(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

Design of a Microprocessor-Based Control System for the Monterey Institute for Research in Astronomy 36" Telescope

by

David Paul Wood

June 1992

Thesis Advisor: W. Bruce Weaver
Co-Advisor I. M. Ross

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b. RESTRICTIVE MARKINGS:

2a. SECURITY CLASSIFICATION AUTHORITY:

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE:

3. DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S):

5. MONITORING ORGANIZATION REPORT NUMBER(S):

6a. NAME OF PERFORMING ORGANIZATION: NAVAL POSTGRADUATE SCHOOL

6b. OFFICE SYMBOL *(If applicable)***:** 39

7a. NAME OF MONITORING ORGANIZATION: NAVAL POSTGRADUATE SCHOOL

6c. ADDRESS *(City, State, and ZIP Code)***:** MONTEREY, CA 93943-5000

7b. ADDRESS *(City, State, and ZIP Code)***:** MONTEREY, CA 93943-5000

8a. NAME OF FUNDING / SPONSORING ORGANIZATION:

8b. OFFICE SYMBOL *(If applicable)***:**

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:

8c. ADDRESS *(City, State, and ZIP Code)***:**

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Design of a Microprocessor-Based Control System for the Monterey Institute for Research in Astronomy 36" Telescope

12. PERSONAL AUTHOR(S): Wood, David P.

13a. TYPE OF REPORT: Engineer's Thesis

13b. TIME COVERED: FROM ______ TO ______

14. DATE OF REPORT *(Year, Month, Day)***:** June 1992

15. PAGE COUNT: 171

16. SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Astronomy, Telescope, MIRA, Computer Control

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The use of computer technology to control large pointing systems can significantly improve performance and reduce human work load. The goal of this thesis was to design software for an inexpensive, yet accurate and efficient control system for the 36-inch reflecting telescope owned and operated by the Monterey Institute for Research in Astronomy. Within this thesis, a computer program is developed to automatically move the telescope to a set of celestial coordinates and track with an accuracy of one-tenth of an arc second for five minutes within 75 degrees of the zenith. Set times are anticipated to be between four and thirty seconds. Corrections are made to celestial coordinates to account for precession, nutation, aberration and atmospheric refraction effects. The user is provided an interface to the computer-based system that allows storage and editing of 100 star positions, editing of the system parameters and display of the telescope's status. Manual control of the telescope is also permitted at any time. Safety of the telescope structure is the primary concern of system software.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL: I. M. Ross

22b. TELEPHONE *(Include Area Code)***:** (408) 646-2[illegible]16

22c. OFFICE SYMBOL: AA/Ro

DD Form 1473, JUN 86 — *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited.

Design of a Microprocessor-Based Control System for the Monterey Institute for Research in Astronomy 36" Telescope

by

David P. Wood
Lieutenant, United States Navy
B.S., Virginia Military Institute, 1985
M.S., Naval Postgraduate School, 1991

Submitted in partial fulfillment of the requirements for the degree of

AERONAUTICAL AND ASTRONAUTICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
June 1992

Author: ______________________
David P. Wood

Approved by: ______________________
W. Bruce Weaver, Thesis Advisor

____
I.M. Ross, Thesis Co-Advisor

____
Daniel J. Collins, Chairman
Department of Aeronautical and Astronautical Engineering

____
Richard S. Elster
Dean of Instruction

ABSTRACT

The use of computer technology to control large pointing systems can significantly improve performance and reduce human work load. The goal of this thesis was to design software for an inexpensive, yet accurate and efficient control system for the 36-inch reflecting telescope owned and operated by the Monterey Institute for Research in Astronomy. Within this thesis, a computer program is developed to automatically move the telescope to a set of celestial coordinates and track with an accuracy of one-tenth of an arc second for five minutes within 75° of the zenith. Set times are anticipated to be between four and thirty seconds. Corrections are made to celestial coordinates to account for precession, nutation, aberration and atmospheric refraction effects. The user is provided an interface to the computer-based system that allows storage and editing of 100 star positions, editing of the system parameters and display of the telescope's status. Manual control of the telescope is also permitted at any time. Safety of the telescope structure is the primary concern of system software.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author would like to express his sincere appreciation for the assistance and consideration provided by Dr. Bruce Weaver and the staff of the Monterey Institute for Research in Astronomy. Additionally, the guidance provided by Professors Otto Heinz and I.M. Ross, both of the Naval Postgraduate School, was invaluable. This work is dedicated with love to Ms. Carolyn Pearce of Auckland, New Zealand.

I. INTRODUCTION

Digital computing equipment, in the form of personal computers, can provide a powerful tool to control large physical systems. This thesis explored the feasibility of upgrading the control system of one such system, an astronomical telescope owned by the Monterey Institute for Research in Astronomy (MIRA). A computer program was written to control the telescope's movements for star tracking to several arc seconds degree of accuracy. A tracking rate error of less than a tenth of an arc second degree of accuracy for five minutes within 75° of the zenith was required. Additionally, requirements to interface a personal computer to the existing hardware were determined. Priority was given throughout the design process to the safety of the telescope structure.

A. STATEMENT OF PROBLEM

The MIRA telescope was to be given the capability to meet the following design criteria:

- Given a known star position, slew on command at the maximum acceptable slew rate to within one degree of the star's calculated position.
- Once within one degree of the calculated position, set via the right ascension and declination stepping motors to within one step of the calculated position.

- The star's calculated position is required to be within ten arc seconds of the star's actual apparent position.

- Manual adjustments to eliminate error between actual apparent and calculated positions are to be made via existing hand paddles or the computer keyboard.

- Calculated corrections to a star's tabulated position are to include precession, nutation, aberration and refraction of light through the Earth's atmosphere.

- Once error has been sufficiently reduced, track the star's apparent position by compensating for rotation of the Earth and estimated refraction of the atmosphere.

- A user interface is required, using standard astronomical vocabulary and symbols. This interface shall include the capability to read, write and edit a listing of one hundred star positions from a formatted data file.

- The telescope's status is to be displayed at all times during telescope movement. The status display is to provide the following information: The telescope's position in celestial and terrestrial coordinates, local, sidereal and universal times and the state of each motor.

- The user interface shall also display adequate error and warning messages to inform the user of special circumstances.

- Allow guiding of the telescope manually by hand paddles or keyboard at any time to eliminate or reduce subsequent errors.

Use of a digital computer was chosen as the simplest design solution. Motor controller hardware, telescope position data and user choices were to be interfaced to this computer via the computer program developed by this thesis.

B. THESIS SCOPE

Requirements to upgrade existing MIRA hardware to encompass an automatic control system for the telescope fell into two categories; hardware and software. This thesis

determined the hardware required to interface the existing motor sets and position determination devices with a personal computer and developed the software which that computer would use to control motor movement. The tracking accuracy requirement of one-tenth of an arc second required that the relation of the Earth's movement to any target star be precisely determined and accounted for. Refraction of light through a variable atmosphere and aberration effects due to the Earth's motion were also considered.

The concept of a virtual and a physical telescope is used throughout this thesis. The virtual telescope is an abstraction in software upon which operations are performed without effecting the movement of physical objects. The virtual telescope represents ideal viewing conditions, a stable Earth with no atmosphere and no Sun to perturb the path of incoming light. Coordinates for a star, corrected only for precession, define the desired position of the virtual telescope for a given time. This position is easily calculated by an astronomer and is used to interface with the computer system. In reality, however, many sources of error other than precession must be considered, such as nutation, aberration and refraction of light through the Earth's atmosphere. The correction of a star's coordinates for all of these errors defines the required position of the physical telescope. An astronomer, in effect, positions the virtual telescope based on incomplete, yet readily obtainable data.

The control system, guided by the computer, corrects the pointing instructions and positions the physical telescope. In effect, the virtual telescope is defined as pointing to a star's correct geometrical coordinates. The physical telescope must point elsewhere in order to see an object at those coordinates.

Corrections to the virtual telescope include precession, nutation, aberration, and atmospheric refraction effects. Structural considerations such as telescope flexure, fork twist, droop, and encoder eccentricity also cause errors which were not considered within this thesis. These corrections require careful measurements of the physical telescope and must be made prior to system completion.

To prevent distortion, the telescope's main mirror is only rear and side defined; it is not front constrained in its housing. Were the telescope to become inverted, this mirror would lose alignment and even possibly be at risk. Safety of this mirror was the overriding design concern.

C. BACKGROUND

1. THE MONTEREY INSTITUTE FOR RESEARCH IN ASTRONOMY

The Monterey Institute for Research in Astronomy is the only privately-owned professional astronomical observatory in the United States to be founded in this century. MIRA is a non-profit corporation dedicated to research and education in astronomy. It maintains a close association with two

institutions of higher learning in Monterey, the Monterey Peninsula College and the U.S. Naval Postgraduate School (NPS). A memorandum of understanding exists between MIRA and NPS allowing the use of government-owned equipment on MIRA property and the teaching of NPS students by MIRA astronomers.

MIRA owns and operates a 36-inch reflecting telescope located on Chews Ridge in the Los Padres National Forest. The Oliver Observing Station is located at approximately 5000 feet altitude above sea level on land owned by the U.S. Forest Service. Additionally, facilities in Monterey include administrative offices and an engineering and receiving building with computer, electronics and machining facilities.

2. CURRENT HARDWARE CONFIGURATION

The telescope at Chews Ridge is a 36-inch reflecting telescope. A 36-inch f/4 paraboloid mirror reflects incident light to a second, smaller cassegrain hyperboloid mirror mounted axially in front of it (see Figure 1). This second mirror directs light to the Guide and Acquisition Package (GAP). The telescope has a back focus of fourteen inches. The GAP is mounted axially, with a zero to six-inch back focus and may contain many instruments, including a spectrometer, a CCD camera, photographic equipment, and an optical eyepiece. The entire telescope is mounted equatorially, which means that it rotates about an axis parallel to the Earth's axis.

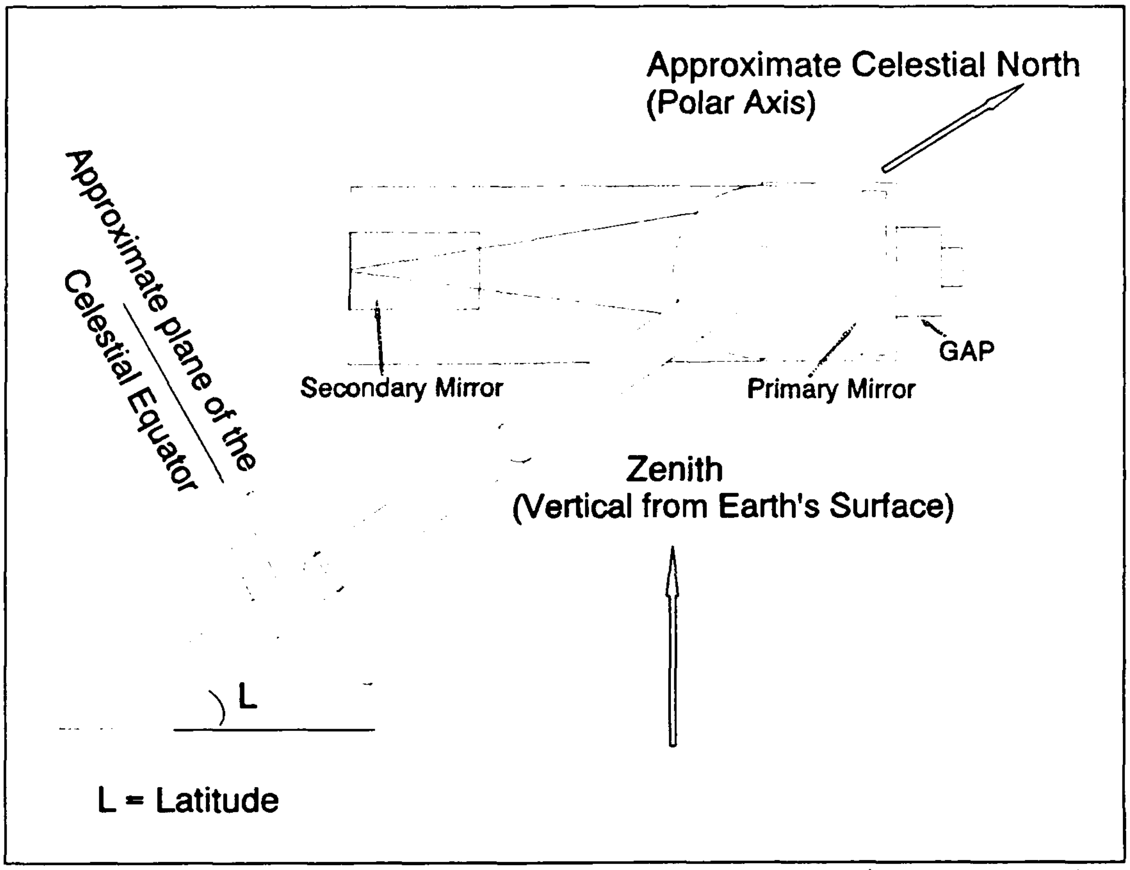

Figure 1 - The MIRA 36" reflecting telescope with equatorial mount

Rotation about the polar axis is then equivalent to changing the right ascension of the telescope (see Chapter II). Rotation about an axis orthogonal to the polar axis is also possible, which changes the telescope's declination, or celestial latitude. Two sets of motors are used to control telescope movement. Two one-quarter horse power slew motors, one each for right ascension and declination, are used for large telescope movements. For finer movements, such as tracking a star, two additional stepping motors are used.

An optical encoder on each of the two axes detect movements of the telescope. This information, once referenced to a known location, is decoded by existing electronics and sent via an RS-232 link to provide the telescope's current (uncorrected) celestial coordinates in the form of declination and local hour angle. Right ascension must be calculated from the local hour angle by knowing the current sidereal time.

3. CURRENT PROCEDURES

Prior to this project, the four motors were controlled electro-mechanically by means of switches mounted on hand paddles. Tracking the star, that is, adjusting the right

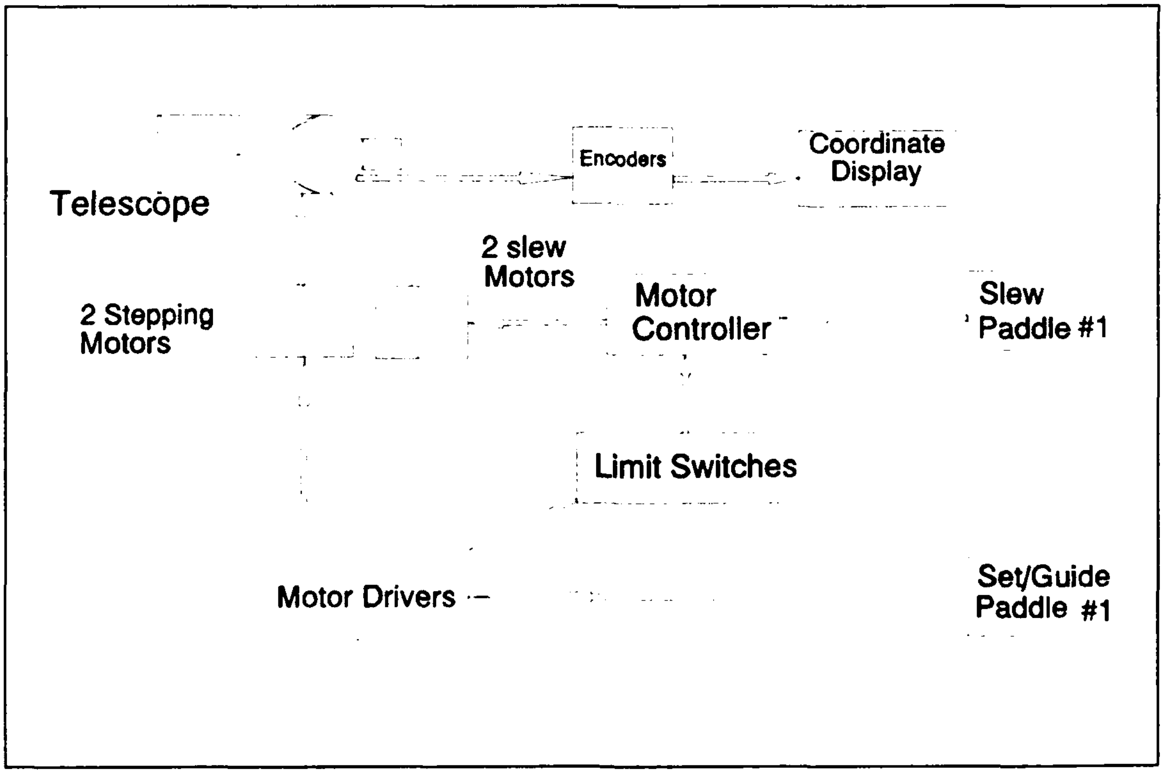

Figure 2 - Existing hardware prior to this thesis

ascension of the telescope at a rate approximately equal to the rotation rate of the Earth, is automatic. Guiding the telescope entails making adjustments for effects not associated solely with the Earth's rotation, such as atmospheric refraction. In this case, the virtual and physical telescopes are identical. An astronomer wishing to observe a star would keep the telescope properly pointed by continual minute adjustments to the stepping motors. This process was long, tedious and tiring which lead directly to the need for an automatic system. Upon completion of the required hardware, astronomers of the future shall be able to sit in a warm-room environment while concentrating on an experiment's data instead of telescope operation.

II. COORDINATE SYSTEMS

A. CELESTIAL COORDINATES

The celestial coordinate system used is centered on the Earth and extends outward to the celestial sphere. The celestial sphere is a hypothetical spherical surface upon which all stars are said to reside. Celestial North is aligned with the Earth's North Pole. Declination corresponds to latitude on the celestial sphere and right ascension is analogous to terrestrial longitude. Where terrestrial longitude is referenced to the prime meridian at Greenwich, England, right ascension is referenced from the prime celestial meridian. The prime celestial meridian passes through the vernal equinox, sometimes called the First Point of Aries. Figures 3, 4 and 5 show these relations.

Right ascension is defined as the angle, measured eastward along the celestial equator, between the vernal equinox and the great circle passing through the poles and the body. Declination is the angle, measured north- or southward along the meridian, from the celestial equator to the body. North declination is taken as positive, south as negative.

An hour circle is a great circle passing through the celestial poles. An hour circle passing through an observer's zenith is also called a celestial meridian. Hour angles are

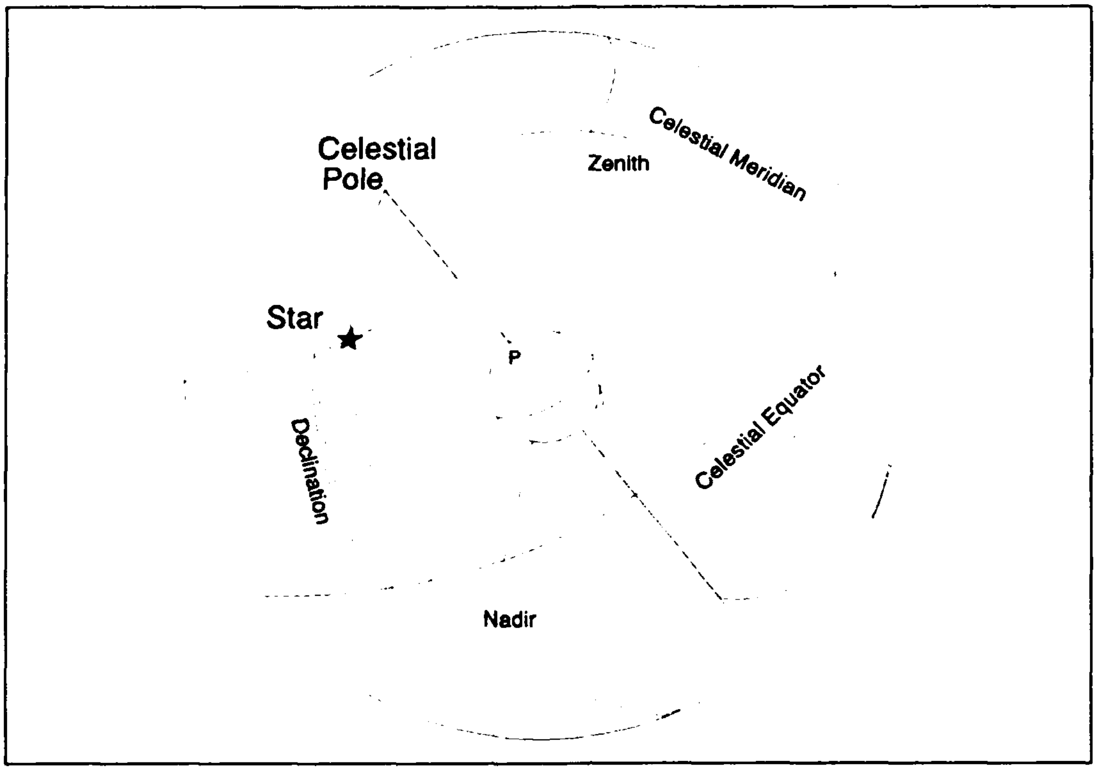

Figure 3 - The celestial sphere

angles measured east- or westward from a celestial meridian to a body. Local hour angles are referenced from an observer's meridian to a body, with westward angles being positive and eastward angles being negative.

From any point on Earth, the point directly overhead is known as the zenith. Its opposite point, directly through the Earth is the nadir. An angle from the horizon to a body is known as the elevation. The zenith angle of a body is simply the angle from the zenith to the position of the body on the celestial sphere. It may be recognized as the co-elevation, or 90 degrees minus the elevation angle. The azimuth of a body is the angle from local North, measured eastward in a

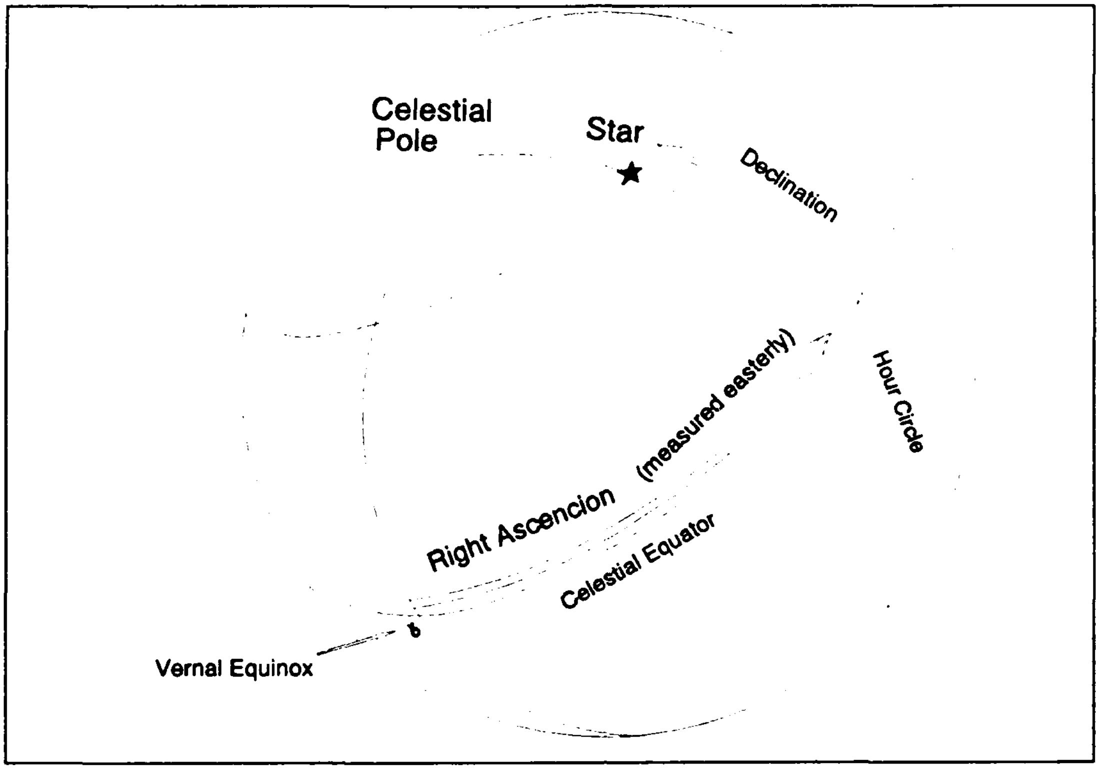

Figure 4 - Right Ascension and Declination

circle, to the body. A body lying due west of an observer would therefore have an azimuth angle of 270 degrees.

B. CONVERSION OF CELESTIAL TO TERRESTRIAL COORDINATES

Safety of the telescope was the primary design criterion. For alignment and safety reasons the telescope could not be pointed below an elevation of 15°, even during testing of the new system. Therefore, care is taken to predict the position in which the telescope would be during a movement. To accomplish this, the celestial coordinates of the active star are converted to their respective terrestrial coordinates, in the form of azimuth and zenith angles. The two coordinate

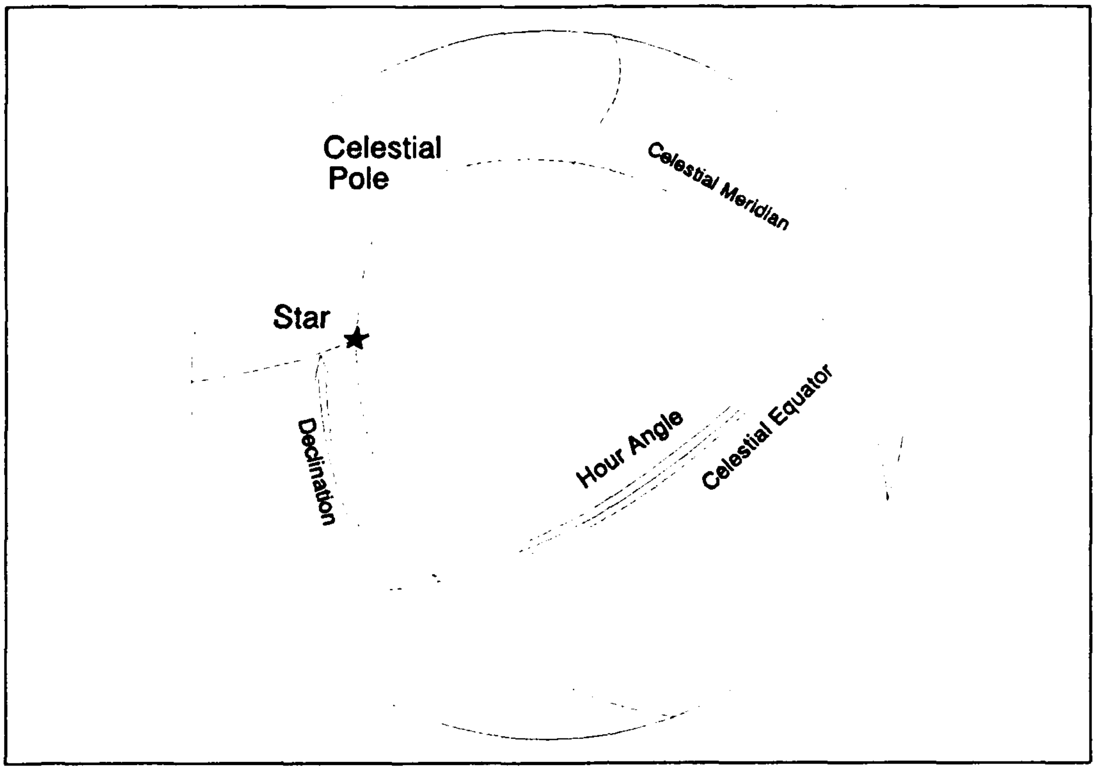

Figure 5 - Hour Angle and Declination

systems are related through a direction cosine approach as follows:

A rectangular coordinate system is determined on the Earth's surface, at the latitude (L) and longitude (γ) of the observing site. This coordinate system is designated by the letter A. Local north is aligned with the number one axis, the number three axis points to the zenith and the number two axis completes a right-handed orthogonal set. Figure 6 shows this arrangement.

The A-frame is related to an intermediate frame, the B-frame, as shown in Figure 6. The number three axis of the B-

frame points to celestial North. This relationship is shown in direction cosine notation in Equation 2.

The B-frame is related to another intermediate frame, denoted by the letter C, by Equation 3. The angle between the number one axes of frames B and C is equal to the longitude of the observing site, measured westerly from Greenwhich.

The final, celestial, frame of reference is designated the D-frame. It is related to the C-frame by the Greenwhich Hour Angle of Aries, which is the angle between number one axes. The Greenwhich Hour Angle of Aries is denoted as GHAγ. This rotation is shown in direction cosine notation in Equation 4.

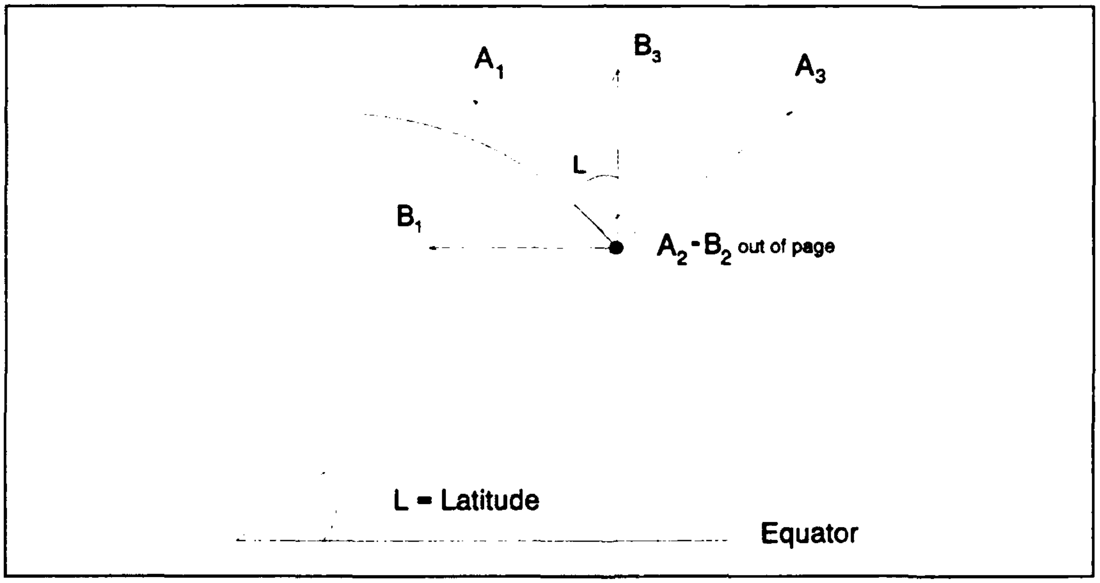

Figure 6 - Definition of the A and B Frames

$$
{}^{D}C^{B}C^{A}=\begin{bmatrix}\sin(L) & 0 & -\cos(L)\\ 0 & 1 & 0\\ \cos(L) & 0 & \sin(L)\end{bmatrix}=[\overline{B_i}\cdot\overline{A_i}] \tag{2}
$$

$$
{}^{D}C^{C}C^{B}=\begin{bmatrix}-\cos(\lambda) & -\sin(\lambda) & 0\\ \sin(\lambda) & -\cos(\lambda) & 0\\ 0 & 0 & 1\end{bmatrix}=[\overline{C_i}\cdot\overline{B_i}] \tag{3}
$$

$$
{}^{D}D^{D}C^{C}=\begin{bmatrix}\cos(GHA\gamma) & -\sin(GHA\gamma) & 0\\ \sin(GHA\gamma) & \cos(GHA\gamma) & 0\\ 0 & 0 & 1\end{bmatrix}=[\overline{D_i}\cdot\overline{C_i}] \tag{4}
$$

$$
\begin{aligned}\overline{d}&={}^{D}C^{C}\,{}^{C}C^{B}\,{}^{B}C^{A}\overline{a}\\ &={}^{D}C^{A}\overline{a}\end{aligned} \tag{5}
$$

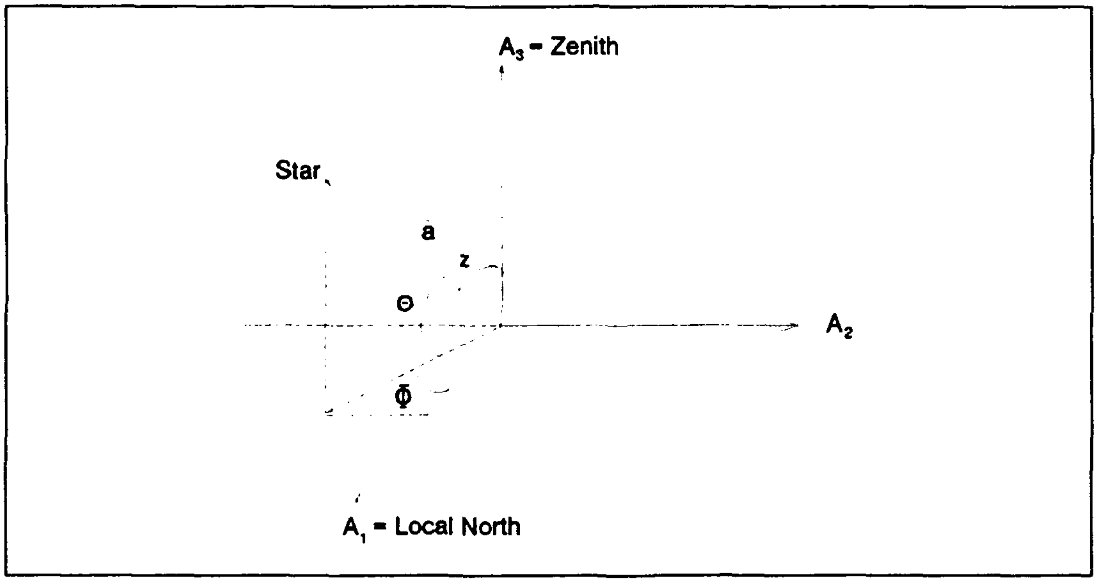

Figure 7 - Azimuth (Φ), Elevation (Θ) and Zenith Angle (z = 90° - Θ) related to rectangular coordinates

The rectangular coordinate vectors must be represented in some more useful form. The unit terrestrial coordinate vector

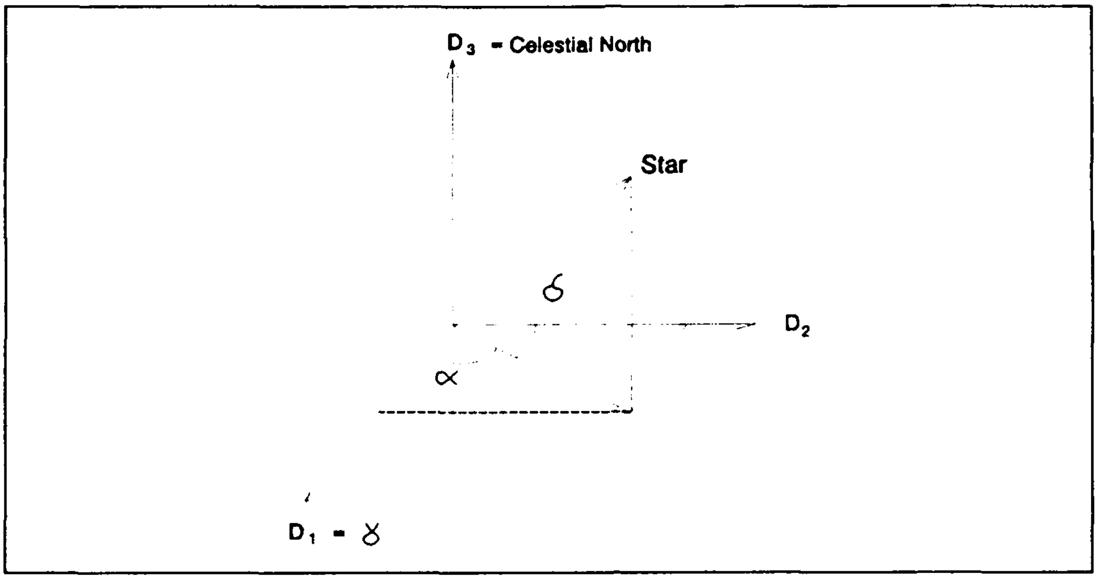

Figure 8 - Right Ascension (α) and Declination (δ) related to celestial rectangular coordinates

that points to a star, a, may be related to azimuth and elevation (or zenith angle) as shown in Figure 7. Note that azimuth is measured clockwise from the A_1 axis and the zenith angle is equal to the co-elevation. Specifically,

$$\bar{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta)\cos(\phi) \\ -\cos(\theta)\sin(\phi) \\ \sin(\theta) \end{bmatrix} \tag{6}$$

Similarly, the celestial coordinate vector may be broken into right ascension and declination as shown in Figure 8 and below.

$$\bar{d}=\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} \cos(\delta)\cos(\alpha) \\ \cos(\delta)\sin(\alpha) \\ \sin(\delta) \end{bmatrix} \tag{7}$$

Solving for the terrestrial coordinates, assuming that α and δ are known, and noting that ${}^{D}C^{A} = [{}^{A}C^{D}]^{-1}$ for an orthogonal set,

$$\bar{a}={}^{A}C^{D}\bar{d} \tag{8}$$

When solving for azimuth, quadrant dependence of the answer must be taken into account. This solution assumes that the distance between the center of the Earth and the observing site is negligable when compared to the distance from the observing site to an observed body. This is true for all observations of stars, but would not be acceptable for viewing closer objects, such as Earth-orbiting satellites.

III. TIME

A. UNIVERSAL COORDINATED TIME

Universal coordinated time (UTC) is the time standard for the National Bureau of Standards and is based on the solar day. This standard of time is based on the atomic second, which is defined as "the duration of 9,192,631,770 periods of the radiation corresponding to the transition between two hyperfine levels of the ground state of the cesium atom 133" (Reference 2, Volume 1) , and is measured by a cesium beam atomic clock. In the United States, radio stations WWV (Fort Collins, CO) and WWVH (Kauai, HI) broadcast time ticks based on UTC at one-second intervals on a continuous basis. The fine timing requirements inherent in astronomical tracking make it desirable to know UTC directly. This was accomplished by reception of the WWV radio signal and decoding into time information.

B. TIME SYSTEM CONVERSION

Universal coordinated time (UTC) may be easily obtained by radio signal. Earth's rotation makes any local coordinate system a moving frame of reference. Since Earth makes one revolution in one sidereal day, this leads naturally to the use of sidereal time for astronomical calculations. Conversion between the two systems is therefore necessary.

The Greenwich mean sidereal time (GMST) is the sidereal time at Greenwich, England as referenced from the mean vernal equinox. Therefore, GMST is also the Greenwich hour angle of the mean equinox. At midnight (0 hour) on any given day GMST may be determined by the following relation:

$$\begin{aligned} GMST\ at\ 0\ hour &= 24110.54841^{s} + 8640184.812866^{s}T_U \\ &+ 0.093104^{s}T_U^2 - 6.2\times10^{-6}T_U^3 \\ T_U &= \frac{(Julian\ Date - 2451545.0)}{36525} \end{aligned} \qquad (9)$$

T_U is defined as the interval of time since noon Universal Time on 1 January, 2000, expressed in Julian centuries. See Reference 1, page B6.

Greenwich apparent sidereal time (GAST) is found by applying a moderate correction to GMST to account for the difference between the true and mean equinoxes.

In the everyday world, we deal with local time as a function of the time zone in which we are. The time at our location is, therefore, the same as our neighbors. Astronomers and others concerned with precise time measurements, however, view local time much differently. Each longitude on Earth has a different hour angle from Greenwich. Local time is defined as Greenwich time plus east longitude. Therefore, you and your neighbor will only share the same local time if you are directly north or south of one another. This is known as local mean solar time.

Local mean sidereal time is then Greenwich mean sidereal time plus east longitude and local apparent sidereal time is Greenwich apparent sidereal time plus east longitude.

From the above it can be easily seen that the local hour angle, important to many calculations regarding stellar positions, may be determined by subtracting the apparent right ascension from the local apparent sidereal time.

C. TIME SYNCHRONIZATION

The Earth revolves on its axis 360 degrees per sidereal day. At this rotation rate, one tenth of a arc-second would only take six-thousandths of a second. The accuracy required for the control system therefore mandates an extremely accurate method for determining and maintaining time information.

In the United States, two radio stations transmit time standard information on a 10 MHz carrier wave. The stations are operated and maintained by the National Institute of Standards and Technology, which was previously known as the National Bureau of Standards. Radio stations WWV, in Fort Collins, Colorado, and WWVH, on Kauai, Hawaii, broadcast these signals which are synchronized to Universal Coordinated Time by way of the U.S. cesium atomic clock.

The method chosen to allow the computer system time clock to be synchronized to this time standard was an electronic radio receiver/decoder made to fit an IBM-compatible personal

computer. This board is manufactured by Odetics Corporation's Precision Time Division, formerly known as Coordinated Time Link, of Anaheim, California. It integrates an AM radio receiver with digital signal processing electronics which decode the WWV/WWVH signal into time and date information. The board is serviced by an external dipole antenna. An onboard battery and clock increment the computer's system clock between synchronizations.

It should be noted that failure of the CTS-10 onboard battery will cause the computer's system clock to be reset to the CTS-10 factory default (afternoon of 19 December, 1991) until the system has re-synchronized.

D. CORRECTING COORDINATES FOR TIME

Many stars are listed in catalogs which give their celestial coordinates for a certain date and time. These reference dates and times are known as epochs. Epoch is Latin for 'time'. Epochs differ primarily due to precession and nutation in the interval separating them. The plane of the Earth's equator, and hence the celestial equator, varies with precession and nutation. Referencing stars' positions to a standard epoch in effect references them to an inertial frame of reference. The two most common epochs for celestial reference are B1950.0 and J2000.0. As one may expect, they refer to the years 1950 and 2000 A.D. The time of the coordinates is taken as midnight on 0 January of that year.

Failure to account for differing epochs can lead to substantial errors in the celestial coordinates of a body. For example, precession alone will cause an error of approximately seven minutes of time in right ascension and five minutes of arc in declination between 1992 and 2000.

The Astronomical Almanac (Reference 1), updated every year, also lists bright stars, with the coordinates given at one-half of the way through the current year. A bright star listed in the 1992 Astronomical Almanac would then be referenced to epoch 1992.5.

Of the corrections made to correct a set of coordinates for time variance, precession is the one commonly referenced to an epoch. The others, nutation and aberration, may be determined separately, through formulae which are known for a given interval of years. Refraction, obviously, does not change with the Earth's position, but is a function of the atmosphere and local stellar coordinates.

Precession corrections in this thesis to the current epoch, i.e. the current date and time, are made by referencing a star to J2000.0, then back to the current epoch. This assures a commonality of treatment for all stars entered.

IV. CORRECTIONS TO CELESTIAL COORDINATES

The Earth is not a perfect sphere of uniform density (i.e. a centrobaric body). This causes close celestial bodies to exert disturbing gravitational moments on it. Disturbing moments with long periods of oscillation are called precession, those with short periods are called nutation. These moments cause the Earth's pole to "wobble" in relation to the plane of the ecliptic. This motion must be taken into account for precise angular determination of stars.

Movement of the Earth about the Sun causes apparent displacement of a celestial body called annual aberration. At the degree of accuracy required, aberration effects, too, must be used to correct a star's apparent position.

The Earth's atmosphere changes in pressure and temperature regularly, causing changes in the refraction of light incident upon it. Refraction must be modeled and its effects taken into account. For these and other reasons, a star's tabulated position must be modified in order to determine where it will actually appear in the night sky. The sight must be corrected for the phenomena of refraction, precession, aberration and nutation.

Table (1)

Correction	Typical Size	Cause	Residual Error after correction
Precession: - lunisolar - planetary	 < 50 arc"/yr <.09 arc"/yr	Gravitational attraction of celestial bodies	 <.05 arc"/yr N/A
Nutation	< 9.2 arc"	Lunar gravitational attraction	<.09 arc"
Aberration: - annual - diurnal	 < 20.5 arc" < 0.03 arc"	 motion of Earth	 <.03 arc" N/A
Refraction	< 21 arc"	Earth's atmosphere	<.1 arc"
Proper motion of stars	<<.1 arc"	Motion of stars	N/A

Each of these phenomena will be described in detail and the necessary corrections explained in this chapter. Of these corrections, only precession is applied to the virtual

telescope. This is because precession is readily calculable by an astronomer and is commonly calculated for each star position taken from a tabulated reference. Nutation, aberration and refraction are calculation-intensive and are solely the responsibility of the computer. These corrections are applied to the physical telescope, but are not shown to the astronomer on the status screen. The user is thereby insulated from the minutia of the control system and can concentrate on an intuitive understanding of the telescope's position.

Table (1) summarizes corrections to celestial coordinates.

A. PRECESSION

The Astronomical Almanac, Reference 1, defines precession as:

> "the uniformly progressing motion of the pole of a freely rotating body undergoing torque from external gravitational forces. In the case of the Earth, the component of precession caused by the Sun and Moon acting on the Earth's equatorial bulge is called lunisolar precession; the component caused by the action of the planets is called planetary precession. The sum of lunisolar and planetary precession is called general precession."

Precession of the Earth's axis is caused primarily by the effects of lunisolar moments due to the much larger gravitational attractions of those bodies. One precessional period is approximately 26,000 years and accounts for the apparent changing of the "North" star in different centuries.

Gravitational attraction would be symmetric about a perfect sphere of uniform mass distribution and no precession would therefore occur. Since the Earth is not a uniform sphere, attraction to distant bodies varies with density and distribution of mass. On the first order, Earth may be pictured as having a belt, or bulge, about the equator.

Figure 9 shows the "misalignment" of the Earth's North pole with the pole of the ecliptic. The angle ϵ is called the obliquity of the ecliptic and is the angle between the Earth's equatorial bulge and the plane of the ecliptic. Figure 10 shows the effects of such an arrangement. In the figure, the lower part of the ellipsoid would be more stronger attracted to the point O than the upper part due to a smaller distance OP. This causes a moment counterclockwise acting on the body G. This moment would cause a precession of the pole on body G.

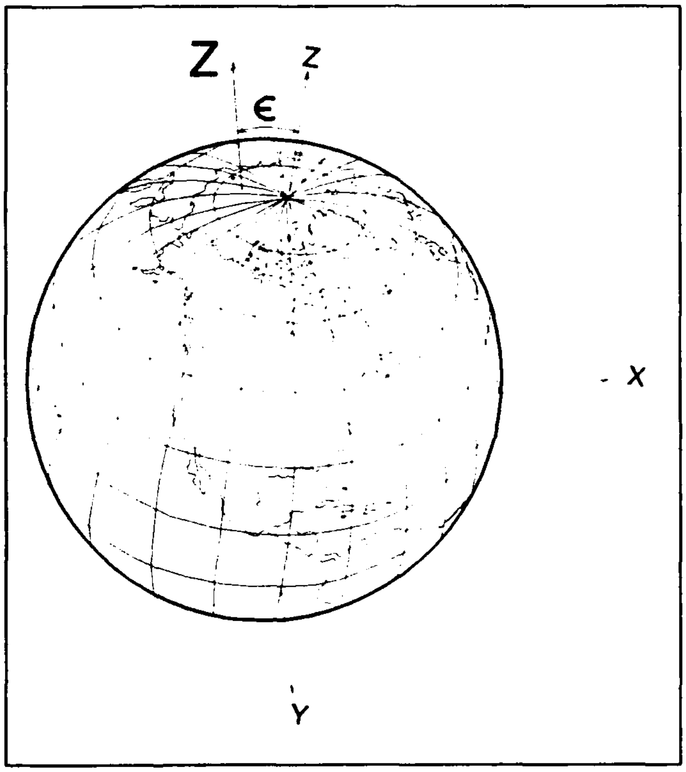

Figure 9 - Effects of nutation on coordinate systems

Precessional effects, although of very long period, are noticeable in astronomical terms. Errors on the order of arc minutes may be observed for every few years of difference from the referenced epoch if precessional effects are not taken into account. Since the MIRA requirement of a few arc seconds accuracy is far below that, precession must be taken into account.

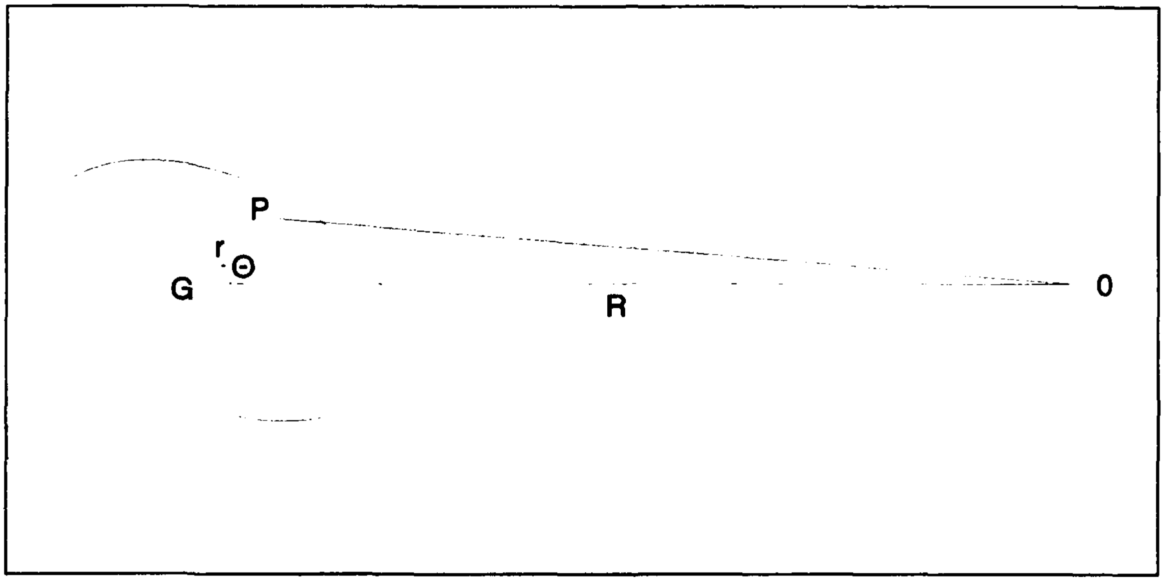

Figure 10 - Causes of nutation

Star data in the Astronomical Almanac is generally referenced to an epoch corresponding to the middle of the current year. Other star catalogs typically represent star coordinates from the standard epochs of B1950.0 or J2000.0 (See Chapter III for an explanation of epochs). Coordinates need, therefore, to be referenced to a known standard, then to the current date for use. The approach used referenced all stars first to the J2000.0 epoch and then to the current date.

Two precessional corrections were then made for each set of star coordinates.

From the Astronomical Almanac, page B19:

For reduction to J2000.0:

$$\alpha_o=\alpha-M-N\sin(\alpha_m)\tan(\delta_m)$$
$$\delta_o=\delta-N\cos(\alpha_m) \tag{10}$$

$$\alpha_m=\alpha-\frac{1}{2}(M+N\sin(\alpha)\tan(\delta))$$
$$\delta_m=\delta-\frac{1}{2}(N\cos(\alpha_M)) \tag{11}$$

For reduction from J2000.0:

$$\alpha_o=\alpha+M+N\sin(\alpha_m)\tan(\delta_m)$$
$$\delta_o=\delta+N\cos(\alpha_m) \tag{12}$$

$$\alpha_m=\alpha+\frac{1}{2}(M+N\sin(\alpha)\tan(\delta))$$
$$\delta_m=\delta+\frac{1}{2}(N\cos(\alpha_M)) \tag{13}$$

where the subscript zero refers to epoch J2000.0 and the subscript m refers to the mean epoch.

The precessional constants M and N are given by:

$$M=1.2812323^{\circ}T+0.0003879^{\circ}T^2+0.0000101^{\circ}T^3$$
$$N=0.5567530^{\circ}T-0.0001185^{\circ}T^2-0.0000116^{\circ}T^3 \tag{14}$$

where T = (Julian Date - 245 1545.0)/36 525

The precessional coefficients of M and N change less than 0.00000005 degrees per year around the J2000.0 epoch. Since it would take approximately 50 years to accumulate errors on the one-tenth arc second order of magnitude, no attempt was made to change the precession coefficients annually.

B. NUTATION

Nutation, or "nodding", is another name for short-period oscillations in the precession moments. For the Earth, approximately 18.5 years describe one nutation period. When compared to the Moon, the Sun has almost no short-period effects and they are normally neglected. The effect of the Moon's gravitational attraction to the Earth's ellipsoidal shape is therefore responsible for the "nodding" of the Earth's pole.

$$\begin{aligned} \Delta\Psi &= -0.0048^\circ \sin(318.5^\circ - 0.053d) \\ &\quad -0.0004^\circ \sin(198.8^\circ + 1.971d) \\ \Delta\epsilon &= +0.0026^\circ \cos(318.5^\circ - 0.053d) \\ &\quad +0.0002^\circ \cos(198.8^\circ + 1.971d) \end{aligned} \tag{15}$$

where d = Julian Date - 244 7891.5

$\epsilon = 23.44^\circ$ = Obliquity of the ecliptic.

The corresponding rotation matrix is

$$N=\begin{bmatrix} 1 & -\Delta\Psi\cos(\epsilon) & -\Delta\Psi\sin(\epsilon) \\ \Delta\Psi\cos(\epsilon) & 1 & -\Delta\epsilon \\ \Delta\Psi\sin(\epsilon) & \Delta\epsilon & 1 \end{bmatrix} \tag{16}$$

The celestial coordinates may be corrected for nutation by multiplying the rotation matrix and the coordinate vector in rectangular form:

$$\overline{S_2}=N\overline{S_1} \tag{17}$$

The new coordinates must then be decoded from rectangular form to right ascension and declination.

C. ABERRATION

Stellar objects may be apparently displaced from their true, geometric positions by the motions of the observer and the object. The finite velocity of light may also cause the apparent location of a body to differ from its actual position. The combination of these effects is called aberration. Of the motions involved, the only one which is significant to the accuracy required is annual aberration. Annual aberration is that component of aberration caused by the motion of the Earth about the Sun.

First order annual aberration corrections to right ascension and declination are given by:

$$\Delta\alpha = \frac{-k\sin(\lambda)\cos(\epsilon)\cos(\alpha)}{\cos(\delta)}$$
$$\Delta\delta = -k\sin(\lambda)\cos(\alpha)\sin(\delta) + k\cos(\lambda)\left[\cos(\epsilon)\sin(\alpha)\sin(\delta) - \sin(\epsilon)\cos(\delta)\right] \quad \textbf{(19)}$$

where λ = true longitude of the Sun, the angle from the First Point of Aries to the Sun, measured eastward in the plane of the ecliptic.

ϵ = obliquity of the ecliptic

and the constant of aberration is given by:

$$k = \frac{2\pi a}{Pc}(1-e^2)^{-\frac{1}{2}} = 20.496 \text{ arc } seconds \quad \textbf{(20)}$$

where a = 1.49598×10^{13} cm = 1 AU

e = 0.01675 = mean eccentricity of the Earth's orbit

P = 3.1558×10^{7} sec = length of sidereal year

c = 2.997925×10^{10} cm/sec = velocity of light

The true longitude of the Sun is given by:

$$\lambda = 278.926^\circ + 0.98567d \quad \textbf{(21)}$$

where d = day of year + fraction of day since 0^h UT

The true longitude of the Sun is an approximation good to about one arc-second. However, since it affects aberration only through a product of trigonometric expressions, the above approximation of aberration is adequate to the one-tenth arc-second level.

D. REFRACTION

When a photon transits from one medium to another in its travels, it refracts. In other words, the direction of its travel is altered. For the case of a photon traveling through space from a distant star and impacting the Earth's atmosphere (a denser medium), the photon is refracted toward a vector normal to the atmosphere. This phenomenon is shown in Figure 11 below and described by Equation 21.

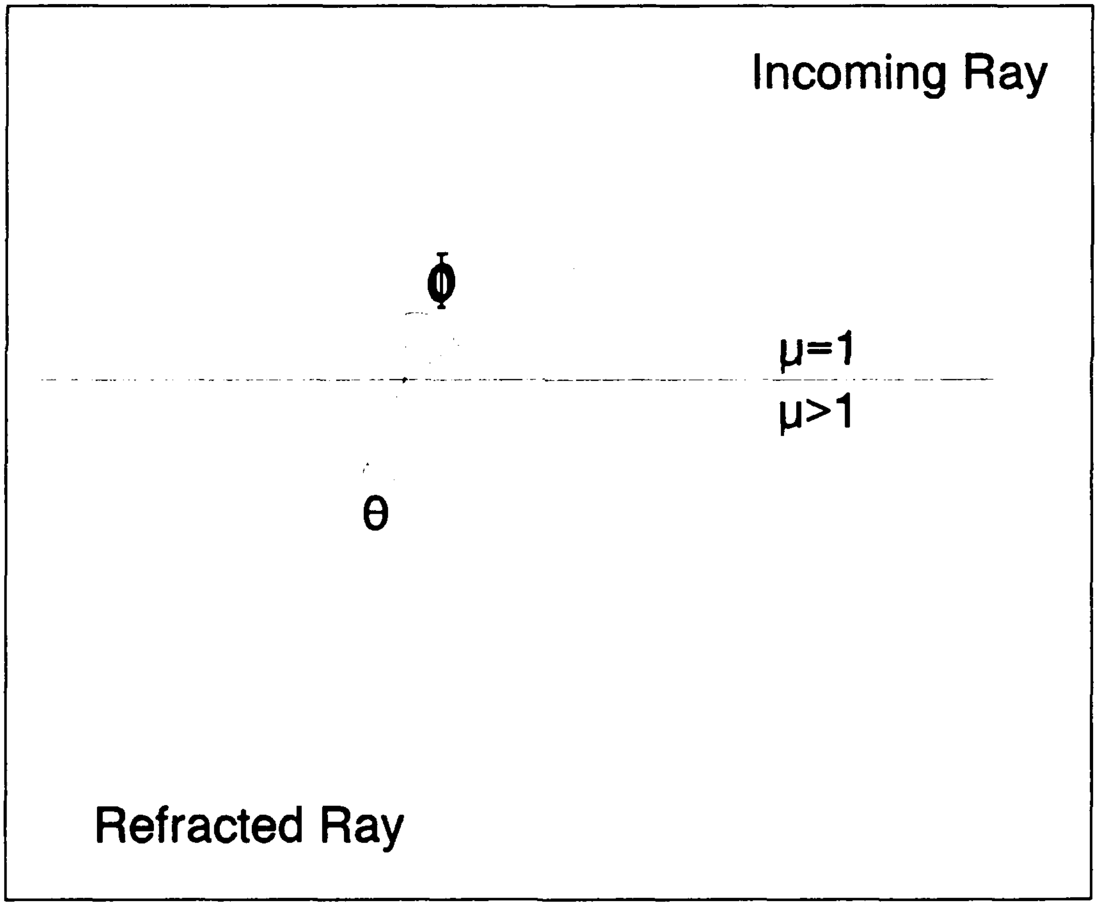

Figure 11 - Refraction through a medium

$$\mu_1 SIN(\Phi) = \mu_2 SIN(\Theta) \tag{22}$$

where $\mu 1$ = The index of refraction of space (= 1),

$\mu 2$ = The index of refraction of the atmosphere,

Φ = the angle of incidence between the incident ray and the atmospheric normal,

Θ = the angle of refraction between the refracted ray and the atmospheric normal.

Since the Earth's atmosphere is constantly changing, refraction is dependent upon many factors and a single index of refraction is not possible to determine. Of these factors, the most prominent are atmospheric pressure and temperature and the wavelength of the observed light.

The MIRA telescope operates with zenith angles of less than 75 degrees for safety of the primary mirror. The following formulae allow a reasonably close approximation of atmospheric refraction for zenith angles less than 75° (more than 15° off the horizon) (See Reference 5).

$$K = \frac{R}{\tan(z_o)} \tag{23}$$

$$R = \frac{21.3'' P\left(1 + 0.\frac{0057}{\lambda^2}\right)\tan(z_o)}{273 + t} \tag{24}$$

$$d\alpha = \frac{K\sec^2(\delta)\sin(H)}{\tan(\delta)\tan(\Phi)+\cos(H)} \tag{25}$$

where α = star's uncorrected right ascension

α_o = observer's right ascension

= GHAγ - observer's longitude

P = atmospheric pressure (mmHg)

t = atmospheric temperature (^{o}C)

λ = center wavelength of star's light (μ)

δ = star's uncorrected declination

H = star's hour angle = $\alpha_o - \alpha$

Φ = observer's latitude

$$d\delta = \frac{K(\tan(\delta)\cos(H)-\tan(\Phi))}{\tan(\delta)\tan(\Phi)+\cos(H)} \tag{26}$$

This treatment of refraction can be extremely sensitive to changes in the atmospheric parameters. The total refraction correction can change on the order of one-tenth of an arc second in less than 1.7 seconds under nominal atmospheric temperature (10^{o}C) and pressure (633 mmHg) at a nearly zero zenith angle. Refraction was therefore updated every one second, with the clock displays. Temperature changes of a single degree Celsius or pressure changes of less than a single millimeter of mercury can alter the refraction corrections by approximately one-tenth of an arc second at nearly nominal atmospheric conditions at a nearly zero zenith angle. While manual entry of these parameters is acceptable

for short period observations, eventual upgrade to automatic temperature and pressure sensing is recommended.

V. HARDWARE

A. SYSTEM OVERVIEW

The hardware to be added to the telescope system is shown in Figure 12 and may be compared to Figure 1, the current hardware configuration.

An IBM-compatible personal computer, based on an Intel 80486 microprocessor, is to be used as the controlling

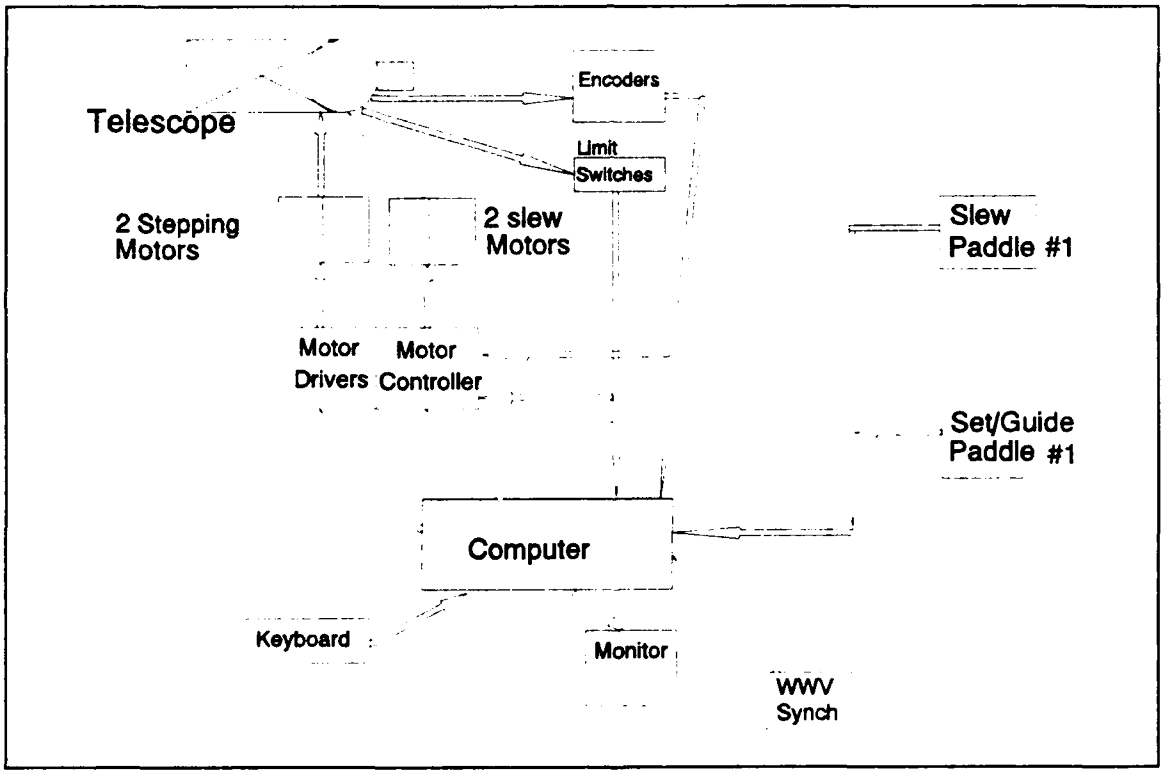

Figure 12 - Proposed hardware configuration

authority. System clock time is to be updated automatically via the WWV time synchronization board discussed in Chapter

III. Slew and set/guide paddles are to be interfaced to the computer via a TTL interface board. Motor drivers are either in an "on" or "off" state. The state is to be controlled by relays operated through an additional TTL interface board.

Physical limit switches are installed to prevent the telescope from being moved beyond its limits in the case of software failure. Two sets of physical limit switches are installed. The first set of switches cuts power to the slew motor controller in the direction required when the telescope's zenith angle reaches 76°. Operation in the opposite direction is allowed to facilitate recovery. This set is engaged first and will stop telescope slew. The second set of limit switches cuts power to the stepping motor drivers, thus preventing the stepping motors from driving the telescope to an unsafe position. In this case, also, only movement in the affected direction is stopped.

B. COMPUTER

An IBM-compatible personal computer was chosen for the system due to availability, cost, speed, and operating considerations. Of these, familiarity of the MIRA staff with the IBM-compatible operating system and the Microsoft QuickBASIC programming language were the key factors. A machine based on an Intel 80486 microprocessor was chosen to allow the system to be operated in a multi-tasking

environment. This eliminated the need for a dedicated computer, cost again being a major consideration.

To allow multi-tasking, the following hardware requirements are recommended as a minimum:

- 80486 microprocessor
- 4 MBytes RAM
- 1 hard disk drive
- 1 floppy disk drive
- CGA monitor
- a mathematics co-processor if a lower CPU is used

If multitasking is not desired, the computer requirements may be downgraded to 1 MByte RAM and an Intel 80286 microprocessor with an 80287 mathematics co-processor.

This type of computer comes standard with eight bus interface slots. Of these, at least five would be needed. One slot would be required for each of the TTL boards for interface to the two control paddles and the two types of motor controllers. One additional slot would be required for the WWV board. Additionally, one RS-232 standard serial interface port would be required for the encoder inputs. This type of port also comes standard with almost any brand of IBM-compatible computer.

VI. PROGRAM DEVELOPMENT

A computer program was developed in the Microsoft QuickBasic Language, version 4.00, to point the telescope to predetermined celestial coordinates and correct for any errors larger than one-tenth of an arc-second. The program uses rotation matrices, developed in Chapter II, to translate between celestial and terrestrial coordinate systems.

The program structure is modular. The main program controls 24 subroutines which deal with data display, data entry, data correction, time correction, key and video handling, and telescope movement. Each subroutine is described below, within its appropriate category.

The concept of a virtual telescope was fundamental to the software development. A 'virtual' model of the system, encompassed solely in software, was developed and the effects of any operation on this model were evaluated prior to any action being taken. This 'virtual' model is seen by the user and represents the target star's position corrected solely for precession. Corrections to raw star position data for precession, nutation, aberration, and refraction were completed and new motor states determined. These corrections and motor states determine the status of the physical telescope. The new position of the physical telescope was determined in both terrestrial and celestial coordinates. If

no errors were encountered, such as hard- or software limits to movement, the telescope was then issued motor commands.

Fig 13 shows the highest-level modular flow of the program.

A. MAIN PROGRAM DESCRIPTION

The main program, beginning on page 1 of Appendix A, controls the flow of telescope operation and includes the main operating loop. A simplified flow chart of the main program is shown in Figure 14.

The program begins by initializing system hardware. It then proceeds to allow user entry of daily atmospheric and observation parameters, if required. The telescope is aligned by pointing to a known bright star. The star's celestial coordinates are entered, thereby giving the system a reference point. Data entry of desired stars for view is accomplished either by direct entry or by reading a previously written and formatted file. Editing of a stored file is allowed. A star is then selected for view. Once a star is selected, the program ensures that the star is visible above the current local horizon. If it is, the main loop is entered, allowing the system to find and track the star. If it is not, another star must be selected.

Since safety of the telescope structure was the primary design criterion, several layers of protection were utilized in the hardware and software. At any time that a zenith angle

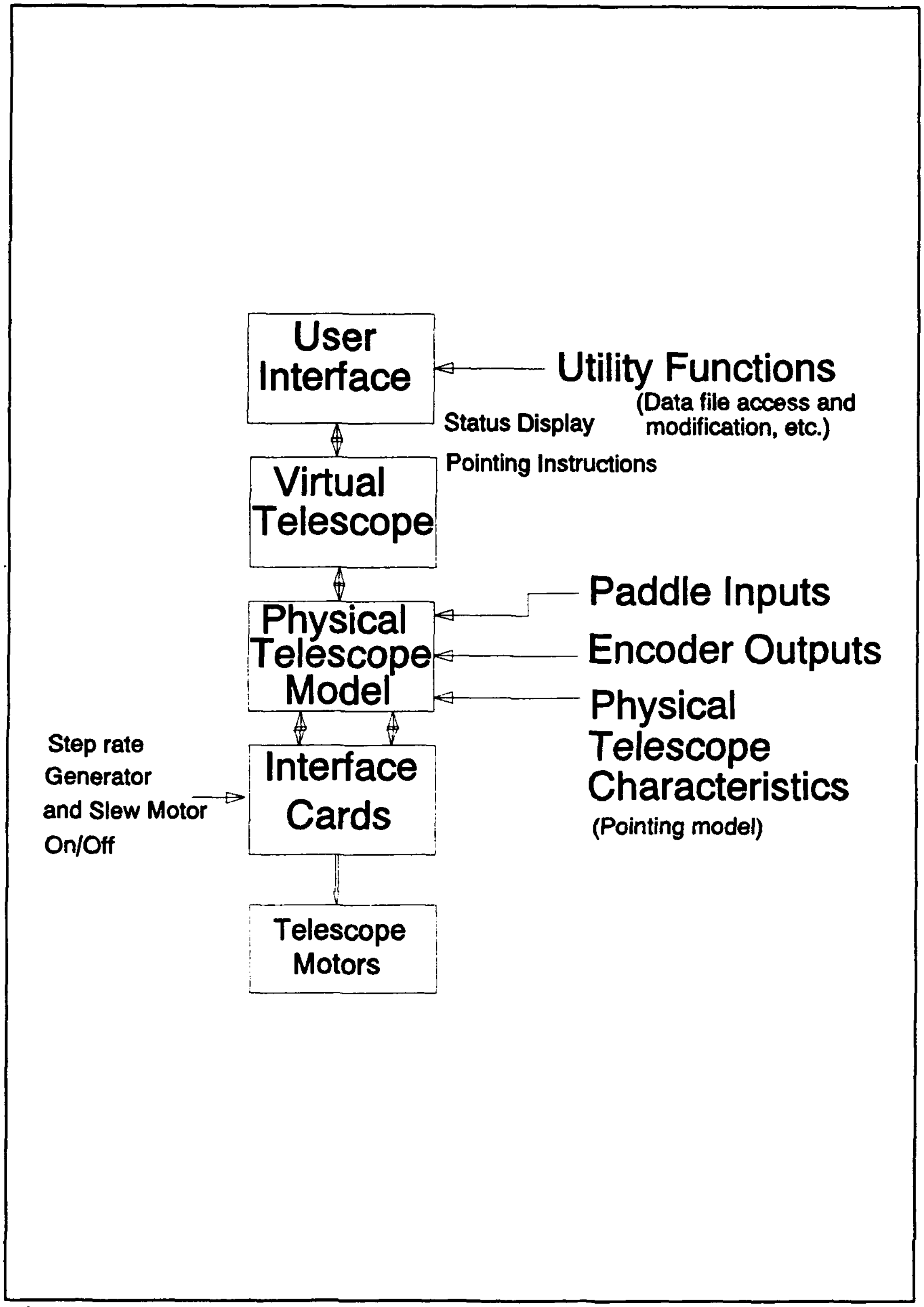

Figure 13 - Program modular concept

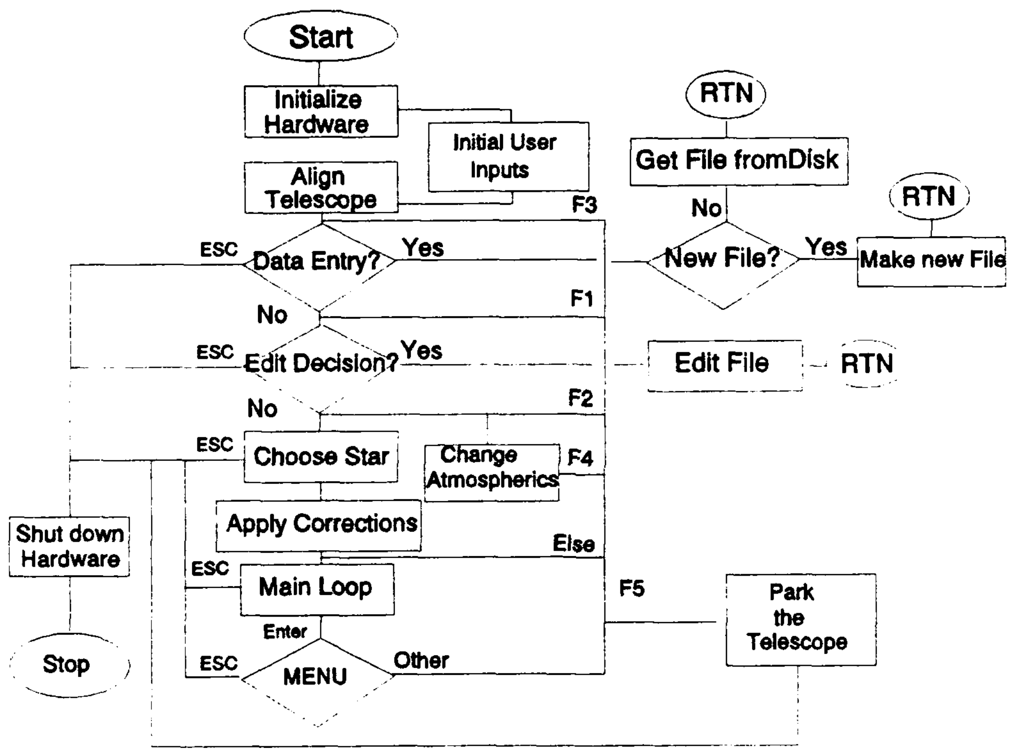

Figure 14 - Simplified flow chart of the main program

of 65 degrees or greater was calculated, a warning message would appear to the operator on the telescope status display screen. When a 75 degree zenith angle was reached, the software would automatically drop track on the target star and ask the operator to choose a new star for view. In the event of catastrophic software errors that allowed the telescope to move beyond a 75 degree zenith angle, physical hardware limit switches were installed at a 76 degree zenith angle. Two sets of limit switches exist on the telescope. The first set cuts power to the slew motor controller. This set of switches must be engaged first due to the speed involved in slewing and the time that the system requires to decelerate. The second set of switches cuts power to the stepping motor drivers.

A simplified flow chart of the main program loop is provided in Figure 15. In this loop the main tracking actions take place. Every second, time and position information is updated to the telescope status screen. The target star's right ascension, declination, hour angle, zenith angle and azimuth are displayed. Movement commands are issued to keep the differences between the stars' coordinates and the telescope's within acceptable limits. The loop also checks once per loop for user input. An ESC key hit stops telescope movement immediately and awaits user confirmation to quit the program. If confirmation is received, the program is exited via a procedure to safely shut down all hardware. If the ENTER key is hit, the change-telescope-status menu appears,

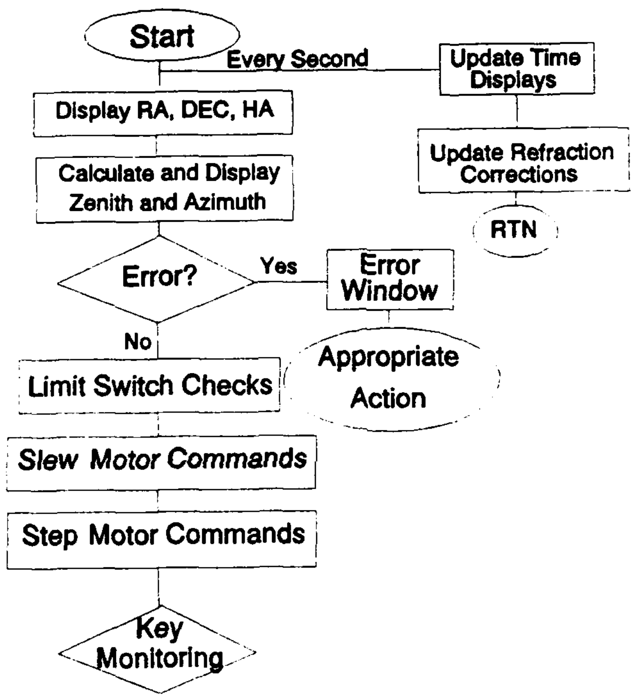

Figure 15 - Simplified flow chart of the main loop

allowing the user several options including the ability to proceed to another star.

Once the virtual telescope's right ascension and declination, the actual telescope's azimuth and zenith angles, the star's hour angle and the times are ready for display in the main loop, the TeleStatusDisplay subroutine is called to display them. It updates the Pacific Standard Time, sidereal time, and universal coordinated time clocks once per second, along with the updated refraction correction.

B. SUBROUTINE DESCRIPTIONS

1. Display Subroutines

Display subroutines are responsible for the display of data and graphics to the computer screen.

a. InitialDisplay

This subroutine creates the display screen graphics which will be utilized for the remainder of the program run. A user box is displayed at the bottom of the screen, in which all user interaction and display occurs. The user box, as shown in Figure 16, is simply a screen area for data exchange. Additionally, menus and error messages occur in windows which occupy other video screens.

This subroutine also colors the background and identifies the program and its intended location to the user.

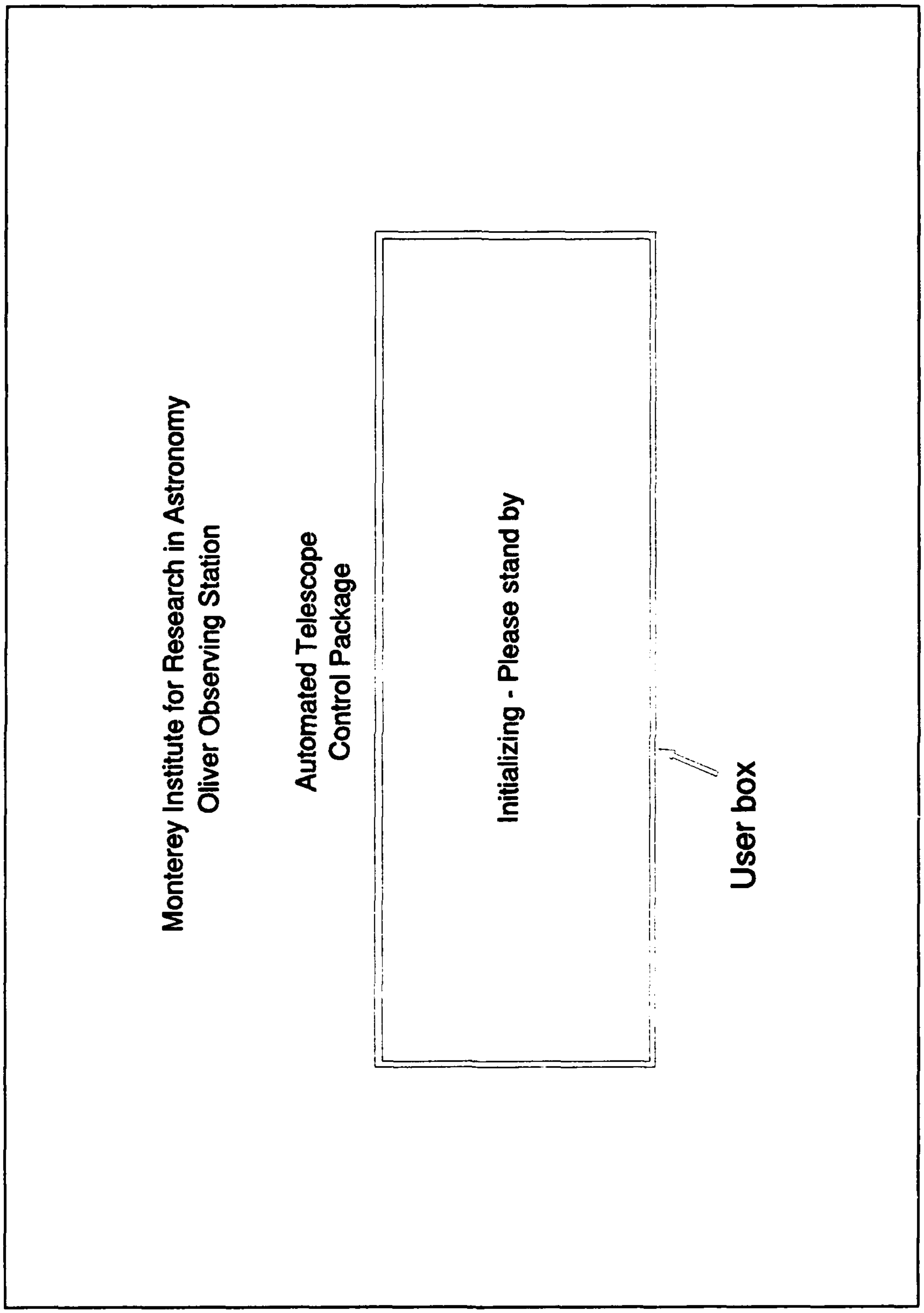

Figure 16 - The initial display screen with user box

b. StarDisplay

This subroutine modifies the initial display screen for star data entry. It draws the individual fields in which the parser (see subroutine StarDataEntry) will operate and also prints instructions for leaving the data entry routine.

c. Windows

All error messages and most menus are displayed on video pages which are separate from the primary display page. These messages and menus are pre-defined data structures and are displayed by this subroutine. A rectangular window is placed at a position determined in the data structure. The size of this window is dependent on the number of lines of text to be displayed and the length of the longest line.

To determine the video parameters and if the next video page is available, the subroutine VideoState is called.

A window is removed by calling the WindowsPop subroutine.

d. WindowsPop

This subroutine removes the last window placed by the subroutine Windows from the active video page and makes the next lowest video page active.

e. TeleStatusDisplay

This subroutine modifies the user box for the display of telescope and star positions and time during the

main loop. The name of the active star is placed in the appropriate box.

f. *ShowTime*

There are three different times displayed on the status screen during telescope movement, universal coordinated time (UTC), mean sidereal time and Pacific Standard time (PST). These three times are updated and displayed through this subroutine.

g. *TeleAngles*

Using the celestial coordinates to which the telescope is pointed, this subroutine determines the position of the telescope in terrestrial terms. The terrestrial rectangular coordinates are decoded into azimuth and zenith angles. Azimuth is defined as the number of degrees measured eastward in a circle from North. North is therefore 0 degrees, East is 90 degrees, etc. Zenith angle is defined as the number of degrees from the point directly overhead the telescope to the telescope pointing vector. Zenith may also be thought of as the co-elevation, where zenith equals 90 degrees minus the elevation. Elevation is the number of degrees from the local horizon to the telescope pointing vector.

The above angles are calculated by means of direction cosine matrices. These matrices are described in detail in Chapter II.

2. Data Entry Subroutines

Data entry subroutines all accept user input prior to proceeding. User input is necessary to align the telescope, to provide the program with atmospheric parameters and a list of stars on which the telescope shall sight.

a. TeleAlign

The telescope must have an alignment point to which all future movements will be referenced. This subroutine accepts user input to determine the right ascension and declination of the telescope pointing vector at a given time. Coordinates from encoder output signals are then treated as a difference from this baseline.

Typically, the right ascension and declination of the present pointing vector will be entered from the bright star list of the Astronomical Almanac. Therefore, the epoch of the coordinates must be given and accounted for. This subroutine precesses, nutates, aberrates and refracts the given coordinates to account for the difference between tabulated and apparent positions of the bright star chosen for alignment.

b. Atmospheric

Refraction through the atmosphere depends primarily on four variables: the zenith angle, the atmospheric temperature, the atmospheric pressure and the center wavelength of light from the star being observed. The latter

three must be provided to the program in order to be accounted for. This subroutine allows user entry of these variables.

c. StarDataEntry

This subroutine is a parser for accepting data for up to one hundred stars. Four fields are required for each listing: star name, right ascension, declination and the epoch to which the right ascension and declination refers. The subroutine displays five lines of the listing at one time and allows scrolling, page jumps and home or end-of-file jumps. Full word-processing ability has been included for appropriate keys. Insert, delete and shift keys are active, with their usual functions.

No attempt was made in this procedure to ensure the correctness of the formatted data. If the data formats are not correct, the program will later display an error message and allow the listing to be edited.

As the listing is updated, the matrix variables Star$(), RA$(), DEC$() and EPOCH$() are updated. These four variables are each one hundred elements in length and hold the star names, right ascensions, declinations and epochs, respectively, in string form.

Figure 17 shows the StarDataEntry screen.

d. ChooseStar

Once a listing of stars has been read into the matrix variables Star$(), RA$(), DEC$() and EPOCH$, one star

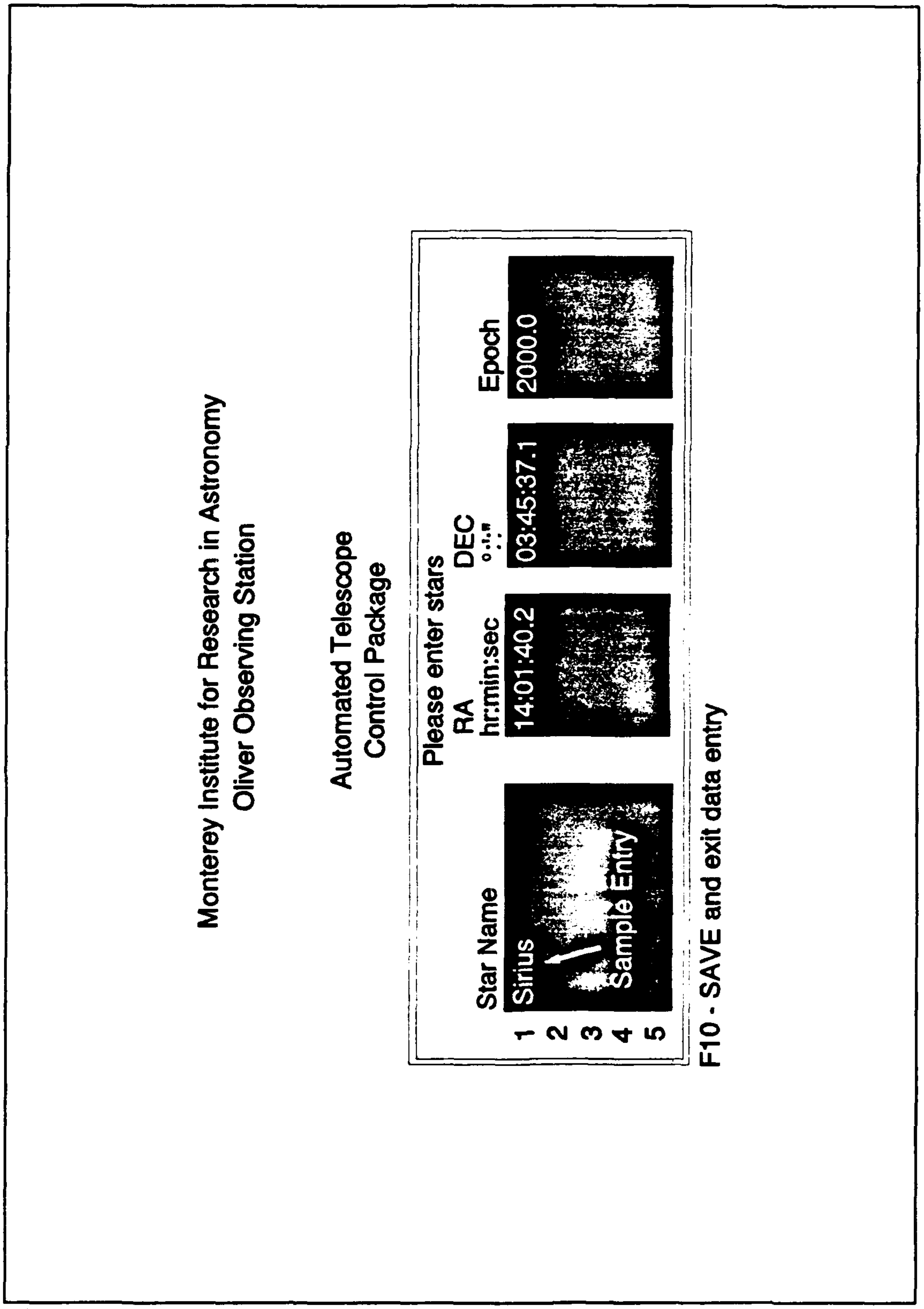

Figure 17 - StarDataEntry screen

must be chosen for view. This is referred to as the "active star". An active star will therefore have associated with it a name, a right ascension, a declination and an epoch.

This subroutine modifies the user box to show a listing of the one hundred star entries in the current file. This file will be on either a floppy or hard disk and may have any file name. The default name is "STARFILE.DAT". The subroutine allows the user to scroll, page up or down, jump home or end-of-file. Once the selected star is highlighted, it is chosen for view by pressing the F10 key. The active star's four elements are then passed to the main loop.

3. Data Correction Subroutines

As described in Chapter IV, the movement of Earth around the Sun, the wobble of Earth's axis, the rotation of the Earth, the relativistic effects of the Sun's presence and the refraction of light through the atmosphere must be accounted for. The data correction subroutines make adjustments to the right ascension and declination of the active star to account for these effects.

a. Precession

The effects of precession on astronomical sightings were discussed in Chapter IV. The Precession subroutine uses the approximate formulae for precession found in the Astronomical Almanac, page B19. The epoch of the active star is first precessed to the standard epoch of J2000.0, then to

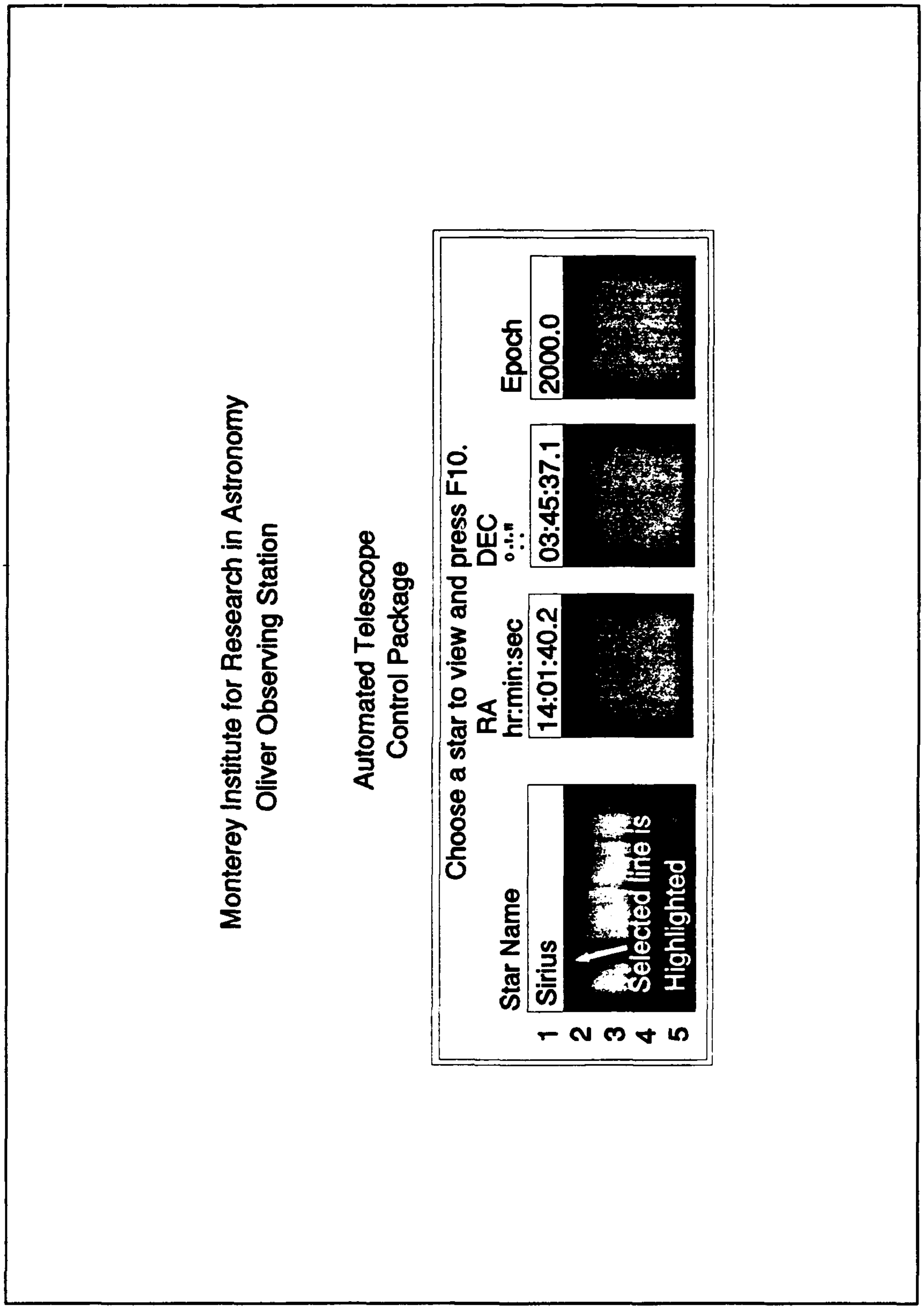

Figure 18 - ChooseStar screen

the current epoch. The current epoch is the calculated to the current day. Right ascension and declination for the current star are then modified to account for the difference between the epoch entered by the user and the current epoch. Precession is accounted for at the beginning of each star sight.

b. Nutation

Nutation is described in Chapter IV. The Nutation subroutine uses the approximate formulae for nutation found in the Astronomical Almanac, page B20. A rotation matrix is used to relate the unmodified and nutated celestial rectangular coordinates. The rectangular coordinates are then decoded into right ascension and declination. Nutation is accounted for at the beginning of each star sight.

c. Aberration

The causes of aberration are briefly discussed in Chapter IV. This subroutine accounts for aberration by determining the relation between the apparent positions of the Sun and the active star and modifying the right ascension and declination accordingly. The method used was garnered from Reference 11, pages 499-502. The mean longitude of the Sun was calculated from formulae provided in the Astronomical Almanac, page C2.

d. Refraction

The effects of refraction through the Earth's atmosphere were considered in Chapter IV. Refraction is affected by four variables: zenith angle, atmospheric temperature, atmospheric pressure and the center wavelength of light for the observation. The latter three were entered by the user in the Atmospheric subroutine. The first was calculated in the TeleAngles subroutine.

Refraction was calculated by the method employed by Reference 5, pages 86-87. Refraction corrections to right ascension and declination were calculated in the subroutine and applied in the main loop. This is necessary because refraction is updated once per second and old corrections must be removed prior to new ones being applied.

4. Time Correction Subroutine

a. SiderealTimeCalc

Sidereal time is a measure of time in relation to the stars (specifically to the vernal equinox), instead of in relation to the Sun. Local mean sidereal time is the actual number of hours, minutes and seconds from the alignment of the local meridian to the mean equinox. This subroutine calculates the local mean sidereal time and adds a minor correction to account for the difference between the true and mean equinoxes. This yields the local apparent sidereal time. This time is then displayed from the main loop.

5. Key and Video Handling Subroutines

a. KeyCode

For any key pressed, KeyCode returns a unique integer value. This value may then be used to determine the key or key combination that has been pressed. Once the subroutine is entered, the program will wait until the key buffer contains a character before returning control to the calling routine. This is especially useful in any application requiring the decoding of user response. The following call KeyCode or KeyCodeNoWait: The main program, ChooseStar, StarDataEntry and Windows.

b. KeyCodeNoWait

KeyCodeNoWait is identical to the KeyCode subroutine above except that the key buffer is checked and control is immediately returned to the calling routine instead of waiting for key input. The main loop uses KeyCodeNoWait to monitor the keyboard while the telescope is tracking a star.

c. VideoState

The video mode parameters of the computer on which the program is being run must be determined in order to place the error message windows and menus. VideoState determines the current video mode, the number of text columns and the number of the active display page. The Windows subroutine uses this data to determine if there is room to place a message window prior to placement.

6. Telescope Movement

Four subroutines control motor commands to the telescope. Each subroutine issues motor commands for a certain type of movement. Movement is categorized into slew, set, track and guide. Slew uses the one-quarter horsepower slew motors to transit the telescope through large angles. This places the telescope in the neighborhood of a target star's position. Set, track and guide all use the smaller stepping motors. Set is used to adjust the telescope's position after slewing and align the telescope's pointing vector to the target star. Once slew and set have occurred, track changes the right ascension of the telescope at a constant rate to account for the Earth's rotation. As refraction, movement and other errors cause misalignments, guide adjusts the telescope's position.

To ensure structure safety, the position of the gravity limit switch is checked after every step of the stepping motors and periodically during use of the slew motors.

Arrow keys are monitored from the main program before each track update to allow manual adjustment of the telescope's position.

a. SlewCommands

The SlewCommands subroutine, immediately after a target star is chosen, slews the telescope to within one

degree of the calculated position. On successive loops, movement is left to the stepping motors in the set (fast) or guide (slow) modes.

The maximum slew motor rate is approximately 12 degrees per second. To ensure the safety of the telescope, the gravity limit switch position is checked every degree of movement or every eight-hundredths of a second during slewing.

b. SetCommands

This subroutine commands the stepping motors following a slew to a new target star. Once the SlewCommands routine has positioned the telescope to within one degree of a calculated position, this subroutine uses the stepping motors, in the fast mode, to bring the position to within one step of the star's calculated position. The astronomer operating the system will then guide the telescope to the star's actual apparent position by operating the stepping motors manually, either by the hand paddles or the arrow keys.

c. TrackCommands

Once a star has been found, the TrackCommands subroutine will update the right ascension motor stepping rate to adjust for changes in the atmospheric refraction. The telescope will, at this time, be slowly sweeping in right ascension to compensate for the rotation of the Earth.

C. ERROR AND WARNING MESSAGES

Error handling is an important aspect of any computer program. To prevent a program from halting catastrophically upon encountering an error, errors must be anticipated and dealt with appropriately. Additionally, some situations may be anticipated that require that the user be warned of impending problems. The main program contains ten error and warning windows to advise the user of anticipated situations.

1. The Quit Confirmation Window

Following standard practice, the ESC key is used to abort a telescope tracking run or to exit the program entirely. Upon striking the ESC key telescope movement is halted, if the telescope is moving. The quit-confirmation window then appears. If the user types either an upper or lower case "Y", the program is terminated. Otherwise, control returns to that segment of the program previously running.

2. The File Save Error Window

When a data file is to be saved, many errors may occur. The appropriate disk drive may not be functioning, the drive may not contain a disk, the disk may be write-protected, the disk may be full, etc. In any of these cases, the file-save-error window appears to inform the user that the data file could not be saved. Upon the press of any key, control returns to the file editor. From this point the user may correct the error or quit the program, as appropriate.

3. The File-Not-Found Error Window

When the user requests that an old data file be used, the program requests the path and file name top retrieve. If the path does not exist or if the file name cannot be located, the file-not-found-error window appears. This window informs the user of the error and waits for any key to be pressed. Control is then returned to the data entry screen menu, allowing the user to choose an old file, re-enter the file name and path, or begin a new file.

4. The Equinox Correction Error Window

Corrections to the Greenwich mean sidereal time are in the form of a number, in seconds. The default values provided are zero seconds for both the current and next days. Should the user fail to except the default values and enter a correction that could not be interpreted as a number in seconds, the equinox-correction-error window appears. once a key is pressed, the user is allowed to except the defaults or re-enter new numbers.

5. The Slew Confirmation Window

Before the telescope is slewed to a new star, the new target star's coordinates are calculated. The zenith angle anticipated is displayed, along with a warning that the F1 key will cause the telescope to slew. Although not displayed on the screen, the F10 key will cause the program to run in

SIMULATION mode. Any other key will return the user to the ChooseStar subroutine.

6. The Change Telescope Status Window

The change-telescope-status window is a user menu of seven possible actions. It appears when the ENTER key is pressed during a telescope tracking run and allows the user to take one of the following actions:

- Edit the current list
- Choose a star from the current list
- Change to a new data file
- Change atmospheric parameters
- Park the telescope
- Exit the menu and return to whatever tracking was previously taking place.

7. The RA/DEC Format Error Window

Right ascension and declination of a target star are entered from a formatted data file. If the program attempts to decode these coordinates and fails, the RA/DEC-format-error window appears. The user is then given the opportunity to edit the data file or choose another star.

8. The Slew Error Window

If the user ignores the slew-confirmation window with a zenith angle over 75 degrees and attempts to slew the telescope to an unsafe position the slew-error window appears. No telescope movement is allowed. Upon the press of any key

control is returned to the change-telescope-status menu so that the user may choose another star or take other appropriate action.

9. The Zenith Angle Warning Window

The maximum zenith angle allowed is 74 degrees. When the zenith angle reaches 75 degrees telescope movement is halted. To warn the user of impending cessation of movement, the zenith-angle-warning window appears any time the zenith angle is 65 degrees or over. It displays the simple warning "The zenith angle is X degrees".

10. The Zenith Angle Halt Movement Window

Should the user ignore the zenith-angle-warning window and allow the telescope to track to its limits, the zenith-angle-halt-movement window appears at 75 degrees. All telescope movement is halted. The change-telescope-status menu appears upon the press of any key. The user may then take action as appropriate such as choosing another star. No action that the user takes can cause the telescope to automatically track to a zenith angle greater than 74 degrees.

11. The Telescope Status Help Window

When the telescope status screen is shown, help for the allowed key strokes is available through this subroutine. The F1 key will display a window listing the keys which control manual slew or guide of the telescope. The arrow keys and numeric keypad is used for this purpose.

12. The CTS-10 Failure Window

It is possible that the CTS-10 WWV time synchronization board will fail and reset. This usually indicates an onboard battery failure or a poor physical connection to the data bus of the computer. In this event, the CTS-10 will reset the computer's system clock to the factory default setting. This condition is tested in the main program upon initialization. If an error is detected, an error window will appear, advising the user to either repair the CTS-10 problem or manually update the system clock to correspond to UTC. After any key is pressed, the program will halt execution and must be run again after the error is rectified.

VII. PROCEDURES

Once final hard- and software modifications have been completed, an astronomer shall have automatic control of telescope movements to a known set of celestial coordinates. He or she will remove the telescope from its stowed configuration by removing obstructing objects, sliding the enclosing roof away from the telescope and opening the doors covering the primary mirror. After this point, procedures differ drastically from the current procedures.

The astronomer will provide power to motor controllers and the computer. Following systems checks, the MIRACTRL program will be run. The user will provide the celestial coordinates to which the telescope is aligned by looking at a known bright star. The astronomer will enter atmospheric data for the evening and provide the computer with a list of star positions either by a data file or by manual entry. He or she will choose a star to view and the program will check whether it is in view. If so, a single key stroke will slew and set the telescope to its desired position. The astronomer will guide the telescope's position onto the star and allow the computer to begin tracking. The computer will continue to track until the astronomer guides the telescope manually, the star is out of view or the gravity limit switch is engaged. The astronomer may choose to halt the telescope, exit the program,

thereby halting the telescope, or change the target star at any time.

The telescope status display screen is shown in Figure 19.

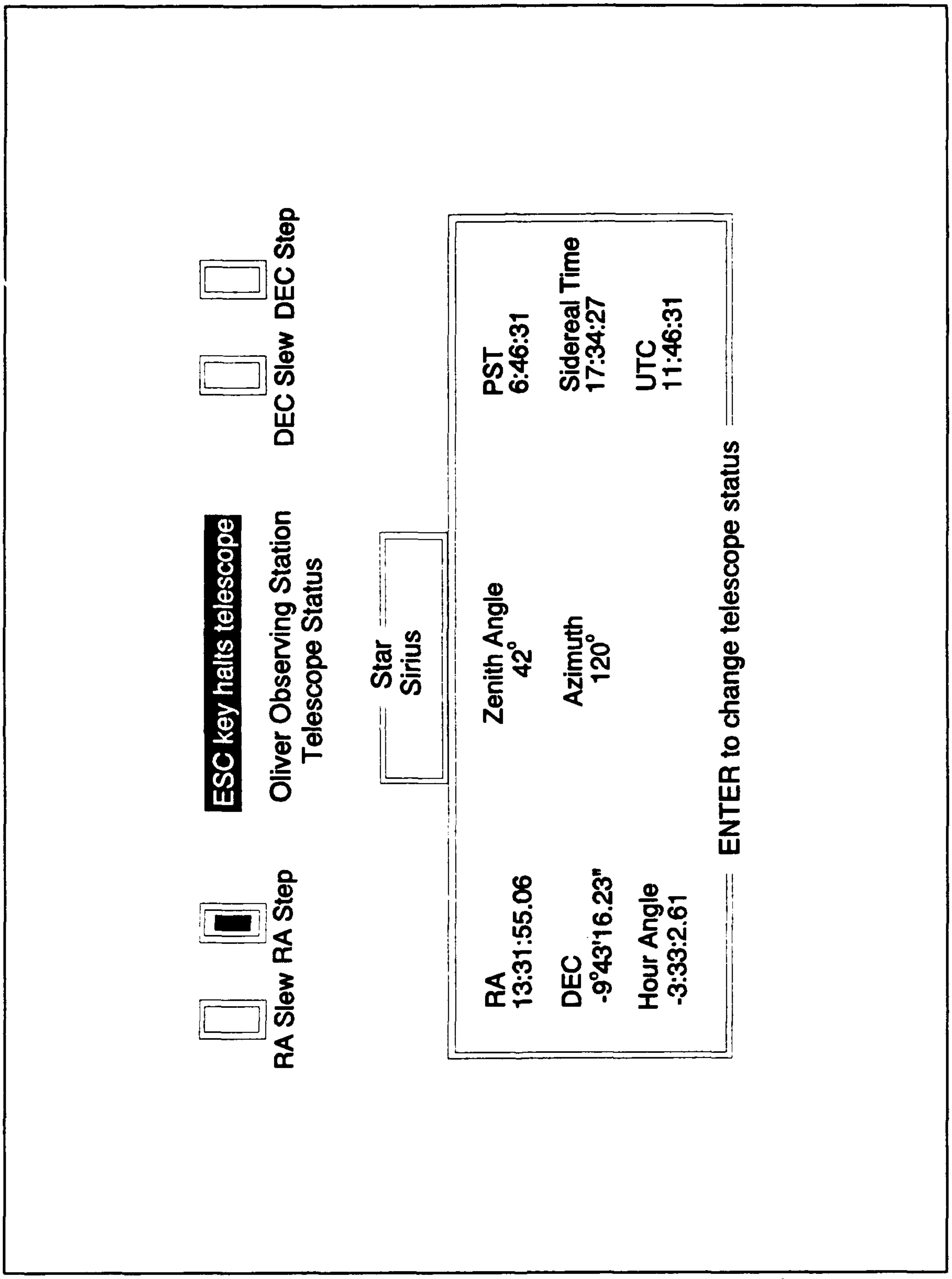

Figure 19 - The telescope status display screen with example data

VIII. PROGRAM VALIDATION

To the date of this writing, the computer program written had not been tested on the telescope due to lack of funds and the time required to connect the associated hardware. Extensive testing of each subroutine that affected telescope movement was accomplished in simulation.

A. TRACKING VALIDATION

To ensure accuracy of the program's tracking commands and related coordinate systems, several simplified simulations were performed. Initially, the latitude of the observing site was set to zero degrees and a star entered with a zero declination. Corrections for precession, nutation, refraction and aberration were not included. This ensured that, with properly defined coordinate systems, the star would have a zenith angle pointing directly East or West at any time. The simulation was run and meridian passage of a simulated star was observed. It was verified that at meridian passage, the local hour angle became zero and switched from a negative to a positive sign. It was also observed that the zenith angle shifted from East to West.

In the next validation simulation, the observing site's latitude was changed to the correct latitude at the Oliver Observing Station. The above simulation was repeated and

meridian passage of the simulated body observed. A target star from the Bright Star listing of the Astronomical Almanac, Reference 1, was then entered and its calculated azimuth and elevation verified by use of a U.S. Navy Rude Starfinder. This process was repeated in each quadrant of right ascension and for both positive and negative declination while varying the system clock to allow for each target star to be in view.

When the system clock was restored to its correct time and the target star was changed to different quadrants, the appropriate error messages were observed when the star was not in view.

B. CELESTIAL COORDINATE CORRECTION VALIDATIONS

The correction of celestial coordinates was verified by allowing only a single correction to be applied at a time. The correction under scrutiny was observed to be within the expected limits set in Table 1. Precession, nutation, aberration and refraction corrections for a given target star were calculated by hand and the results compared to the computer solution. The direction of each correction was verified at concurrently. For example, refraction of light by the Earth's atmosphere always causes the apparent position of a star to be shifted toward the zenith. This was verified in all simulations run.

Corrections to celestial coordinates must take place in the proper order, i.e. precession, nutation, aberration and

refraction. Simulations were also run to verify that the corrections were taking place in the proper sequence and that the only changes to the celestial coordinates came from those sources.

C. MOTOR COMMAND VALIDATION

Motor commands could not be fully validated since the computer was not connected to the motor drivers or controllers. Simulations to verify motor command logic were completed. The telescope status display screen also displays the motor states for all four motors. These displays were used to determine when motors were turned on or off during representative actions. Slew, guide and track motor commands were tested individually and together to ensure that the sequence of commanded motor actions was correct. Following hardware installation, a check should be made of signal output to each motor controller or driver prior to allowing telescope movement. Limit switch actions were also validated by simulation.

IX. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

It has been shown that an automatic telescope control system is both practical and relatively inexpensive. Additional hardware requirements for the system discussed would cost approximately eight thousand dollars as of the time of this writing. While a personal computer based on an Intel 80486 microprocessor was used for this design, it was determined experimentally that the program will run in real time on an Intel 80286-based system, as long as an Intel 80287 mathematics co-processor is installed. This variation would require a dedicated computer, because multitasking on an 80286-based computer is not practical with available operating systems or data bus speeds.

B. RECOMMENDATIONS FOR ADDITIONAL WORK

There remains additional work on the MIRA telescope system before fully automatic operation is possible. Hardware updates yet required include modification of installed optical encoders, replacement of the slew motor drivers and computer control of the focusing step motor.

Completion of computer program subroutines to reflect the types of motor drivers eventually purchased will be necessary. Additionally, a warm-room computer monitor and keyboard as

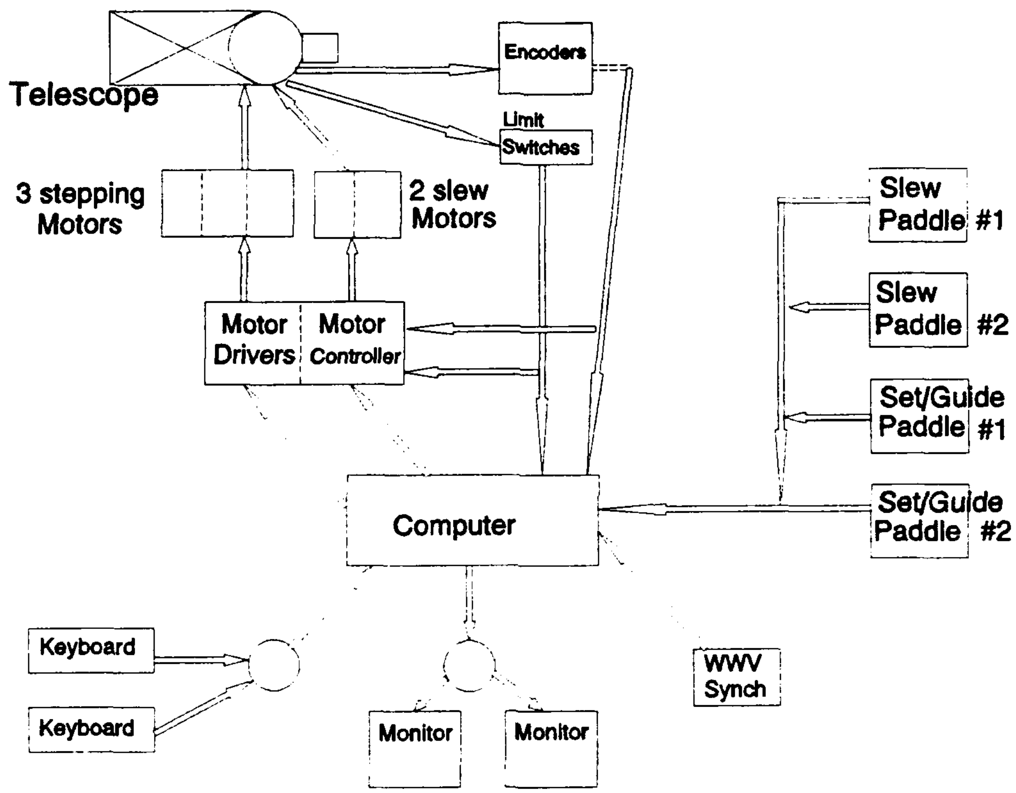

Figure 20 - Eventual hardware configuration

well as slew and guide/track paddles will need to be installed. This work was not completed because of lack of funds. Figure 20 shows the desired hardware configuration upon completion. The pointing model, which represents the changing telescope structure in software, must also be included. Testing of the completed system will require extensive simulation in both hard- and software prior to actual telescope movement. Before the telescope is first moved by computer control, a rigorous testing plan must be developed.

Automatic environment sensing of temperature and pressure is required for long duration tracking of a single body because of refractive effects. Less than one-tenth of an arc second resolution may be obtained under the current model due to manual input of time-varying parameters.

Optimization of slewing movements may be accomplished by a "traveling salesman" approach to minimize the required slew times or accelerations.

APPENDIX A MIRACTRL.EXE

A. MIRACTRL.BAS LISTING

```
'****************************************
'**     Name:       MIRACTRL            **
'**     Type:       Program             **
'**     Module:     MIRACTRL.BAS        **
'**     Language:   Quickbasic 4.50     **
'**     Author:     David P. Wood       **
'****************************************
'
'Control Core for MIRA Telescope.
'
'Usage:          No Command Line Parameters
'.MAK File:      (none)
'Parameters:     (none)
'Variables:      w*Text$()   Text for the error windows
'                 Star$()     A 100-element matrix of desired
stars for
'                            viewing
'                  RA$()       A 100-element matrix of right
ascensions
'                            for the stars listed in Star$()
'                      DEC$()        A 100-element matrix of
declinations
'                            for the stars listed in Star$()
'               Epoch$()    A 100-element matrix of epochs for
the
'                           RA's and DEC's of the stars listed
in Star$()
'                       EQofEQ!()    The Equation of Equinox
corrections to
'                           sidereal time for successive dates
(due
'                              to corrections for the Earth's
nutation
'               EQofEQUIN!  The Equation of Equinox correction
for the
'                            current time
'                 w.*         Windows Type variables, defines
window
'                            appearance
'                min%        The current minute of time
```

```
'                hour%        The current hour of time
'                day%         The current day
'                month%       The current month
'                year%        The current year
'                 LocalTime$  A string representing the local
time
'              SiderealTime$  A string representing the local
'                               apparent sidereal time
'               TIME$        The system clock time string, set
to correspond
'                               to Universal Coordinated time
(UTC)
'                kee%         A unique code for any key struck
on the keyboard
'                              returned by either KeyCode% or
KeyCodeNoWait%
'                             Functions
'               Character$   A string representing a character
returned
'                             from the keyboard
'                Abort%       A variable assigned to a current
keyboard buffer
'                              input used to determine if the
user wishes to
'                             quit the program
'                Quit%        A variable assigned to a current
keyboard buffer
'                              input used to determine if the
Abort% is TRUE
'
'
'Define constants
 CONST ENTER = 13
 CONST TABHIT = 9
 CONST BACKSPACE = 8
 CONST ESCAPE = 27
 CONST SHIFTTAB = 3840
 CONST RIGHTARROW = 19712
 CONST LEFTARROW = 19200
 CONST UPARROW = 18432
 CONST DOWNARROW = 20480
 CONST NUMRIGHTARROW = 54
 CONST NUMLEFTARROW = 52
 CONST NUMUPARROW = 56
 CONST NUMDOWNARROW = 50
 CONST one = 49
 CONST three = 51
 CONST seven = 55
 CONST nine = 57
 CONST F1 = 15104
 CONST F2 = 15360
```

```
CONST F3 = 15616
CONST F4 = 15872
CONST F9 = 17152
CONST F10 = 17408
CONST F11 = -31488
CONST F12 = -31232
CONST DELETE = 21248
CONST FALSE = 0
CONST TRUE = NOT FALSE
CONST PGDOWN = 20736
CONST PGUP = 18688
CONST INSERT = 20992
CONST HOME = 18176
CONST ENDFILE = 20224
CONST PLUS = 1
CONST MINUS = -1
CONST pi = 3.14159265359#
'
'Define color constants
CONST BLACK = 0
CONST BLUE = 1
CONST GREEN = 2
CONST CYAN = 3
CONST RED = 4
CONST MAGENTA = 5
CONST BROWN = 6
CONST WHITE = 7
CONST BRIGHT = 8
CONST BLINK = 16
CONST YELLOW = BROWN + BRIGHT
'
TYPE RegType
    ax            AS INTEGER
    bx            AS INTEGER
    cx            AS INTEGER
    dx            AS INTEGER
    Bp            AS INTEGER
    si            AS INTEGER
    di            AS INTEGER
    flags         AS INTEGER
END TYPE
TYPE RegTypeX
    ax            AS INTEGER
    bx            AS INTEGER
    cx            AS INTEGER
    dx            AS INTEGER
    Bp            AS INTEGER
    si            AS INTEGER
    di            AS INTEGER
    flags         AS INTEGER
    ds            AS INTEGER
```

```
    es              AS INTEGER
END TYPE
'
TYPE WindowsType
    action          AS INTEGER
    edgeline        AS INTEGER
    row             AS INTEGER
    col             AS INTEGER
    fgdEdge         AS INTEGER
    bgdEdge         AS INTEGER
    fgdBody         AS INTEGER
    bgdBody         AS INTEGER
    fgdHighlight    AS INTEGER
    bgdHighlight    AS INTEGER
    fgdTitle        AS INTEGER
    bgdTitle        AS INTEGER
    fgdPrompt       AS INTEGER
    bgdPrompt       AS INTEGER
    returnCode      AS INTEGER
END TYPE
'
'Functions
DECLARE FUNCTION KeyCodeNoWait% (Character$)
DECLARE FUNCTION KeyCode% (Character$)
'
'Subprograms
DECLARE SUB windows (w AS WindowsType, wText$(), wTitle$,
wPrompt$)
DECLARE SUB WindowsPop ()
DECLARE SUB VideoState (mode%, columns%, page%)
DECLARE SUB InitialDisplay ()
DECLARE SUB StarDisplay ()
DECLARE SUB StarDataEntry (Star$(), RA$(), DEC$(), EPOCH$(),
NewFile%, w1 AS WindowsType, w1Text$(), w1Title$, w1Prompt$)
DECLARE SUB ChooseStar (Star$(), RA$(), DEC$(), EPOCH$(),
NewFile%, w1 AS WindowsType, w1Text$(), w1Title$, w1Prompt$,
currentStar%)
DECLARE SUB TeleStatusDisplay (ActiveStar$, Simulation%)
DECLARE SUB DayLightSaving (DayLightSavings%, hour%, day%,
month%, year%)
DECLARE SUB SiderealTimeCalc (SiderealTime$, min%, hour%,
day%, month%, year%, EQofEQUIN!, GMST0hrSEC#, JulianDate#,
dayFraction#, DaysPassed%)
DECLARE SUB EquationEquinoxes (EQofEQ(), EQofEQUIN!)
DECLARE SUB Interrupt (intnum%, inreg AS RegType, outreg AS
RegType)
DECLARE SUB ShowTime (min%, hour%, day%, month%, year%,
DayLightSavings%, SiderealTime$)
DECLARE SUB StarData (Star$(), RA$(), DEC$(), EPOCH$(),
EditDecision%, NewFile%)
```

```
 DECLARE SUB Refraction (Pressure!, Temperature!, Wavelength!,
GMST0hrSEC#, HourAngle#, GHAaries#, RAofStar#, DECofStar#,
RARefractCor#, DecRefractCor#, RAofMIRA#)
 DECLARE SUB Atmospheric (Pressure!, Temperature!,
Wavelength!)
 DECLARE SUB TeleAngles (Zenith%, Azimuth%, GHAaries#,
RAofStar#, DECofStar#, CelCoord#(), DCM#(), TerraRec#())
 DECLARE SUB Precession (RA$(), DEC$(), EPOCH$(),
currentStar%, JulianDate#, EpochError%, RAofStar#, DECofStar#,
RADecError%)
 DECLARE SUB Aberration (RAofStar#, DECofStar#,
RAAberrationCor#, DecAberrationCor#, dayFraction#,
DaysPassed%)
 DECLARE SUB SlewCommands (Simulation%, TeleRA#, TeleDEC#,
RAofStar#, DECofStar#)
 DECLARE SUB TrackCommands ()
 DECLARE SUB SetCommands (Simulation%, TeleRA#, TeleDEC#,
RAofStar#, DECofStar#)
 DECLARE SUB TeleAlign (GMST0hrSEC#, GHAaries#, TeleRA#,
TeleDEC#)
 DECLARE SUB Nutation (RAofStar#, DECofStar#, JulianDate#,
N#(), S1#(), S2#())
'
'Data Structures
 DIM w1 AS WindowsType
 DIM w2 AS WindowsType
 DIM w3 AS WindowsType
 DIM w4 AS WindowsType
 DIM w5 AS WindowsType
 DIM w6 AS WindowsType
 DIM w7 AS WindowsType
 DIM w8 AS WindowsType
 DIM w9 AS WindowsType
 DIM w10 AS WindowsType
 DIM w11 AS WindowsType
 DIM w12 AS WindowsType
'
'Arrays
 DIM w1Text$(1 TO 1)
 DIM w2Text$(1 TO 1)
 DIM w3Text$(1 TO 1)
 DIM w4Text$(1 TO 1)
 DIM w5Text$(1 TO 7)
 DIM w6Text$(1 TO 9)
 DIM w7Text$(1 TO 1)
 DIM w8Text$(1 TO 2)
 DIM w9Text$(1 TO 3)
 DIM w10Text$(1 TO 2)
 DIM w11Text$(1 TO 13)
 DIM w12Text$(1 TO 5)
 DIM Star$(1 TO 100)
```

```
 DIM RA$(1 TO 100)
 DIM DEC$(1 TO 100)
 DIM EPOCH$(1 TO 100)
 DIM EQofEQ!(1 TO 2)
 DIM CelCoord#(1 TO 3)
 DIM DCM#(1 TO 3, 1 TO 3)
 DIM TerraCoord#(1 TO 3)
 DIM TerraRec#(1 TO 3)
 DIM S1#(1 TO 3)
 DIM S2#(1 TO 3)
 DIM N#(1 TO 3, 1 TO 3)
'
'Define error windows
'
'The quit program confirmation window
 w1.action = 1
 w1.edgeline = 1
 w1.row = 5
 w1.col = 5
 w1.fgdEdge = CYAN + BRIGHT
 w1.bgdEdge = RED
 w1.fgdBody = BRIGHT + WHITE
 w1.bgdBody = RED
 w1.fgdHighlight = 0
 w1.bgdHighlight = 0
 w1.fgdTitle = CYAN
 w1.bgdTitle = RED
 w1.fgdPrompt = YELLOW
 w1.bgdPrompt = RED
 w1Title$ = ""
 w1Text$(1) = " Quit Program? (Y or N) "
 w1Prompt$ = ""
'
'The file save error window
 w2.action = 1
 w2.edgeline = 1
 w2.row = 5
 w2.col = 5
 w2.fgdEdge = CYAN + BRIGHT
 w2.bgdEdge = RED
 w2.fgdBody = BRIGHT + WHITE
 w2.bgdBody = RED
 w2.fgdHighlight = 0
 w2.bgdHighlight = 0
 w2.fgdTitle = CYAN
 w2.bgdTitle = RED
 w2.fgdPrompt = YELLOW
 w2.bgdPrompt = RED
 w2Title$ = ""
 w2Text$(1) = "       Save error         "
 w2Prompt$ = "Press any key to continue"
```

```
'
'The file-not-found error window
 w3.action = 1
 w3.edgeline = 1
 w3.row = 5
 w3.col = 5
 w3.fgdEdge = CYAN + BRIGHT
 w3.bgdEdge = RED
 w3.fgdBody = BRIGHT + WHITE
 w3.bgdBody = RED
 w3.fgdHighlight = 0
 w3.bgdHighlight = 0
 w3.fgdTitle = CYAN
 w3.bgdTitle = RED
 w3.fgdPrompt = YELLOW
 w3.bgdPrompt = RED
 w3Title$ = ""
 w3Text$(1) = " Error - File not found or path incorrect "
 w3Prompt$ = "Press any key to continue"
'
'The Equinox correction error window
 w4.action = 1
 w4.edgeline = 1
 w4.row = 5
 w4.col = 5
 w4.fgdEdge = CYAN + BRIGHT
 w4.bgdEdge = RED
 w4.fgdBody = BRIGHT + WHITE
 w4.bgdBody = RED
 w4.fgdHighlight = 0
 w4.bgdHighlight = 0
 w4.fgdTitle = CYAN
 w4.bgdTitle = RED
 w4.fgdPrompt = YELLOW
 w4.bgdPrompt = RED
 w4Title$ = ""
 w4Text$(1) = " Error - Equinox corrections must be a NUMBER
in seconds "
 w4Prompt$ = "Press any key to continue"
'
'The slew confirmation window
 w5.action = 1
 w5.edgeline = 1
 w5.row = 5
 w5.col = 5
 w5.fgdEdge = CYAN + BRIGHT
 w5.bgdEdge = RED
 w5.fgdBody = BRIGHT + WHITE
 w5.bgdBody = RED
 w5.fgdHighlight = 0
 w5.bgdHighlight = 0
```

```
 w5.fgdTitle = WHITE + BLINK + BRIGHT
 w5.bgdTitle = RED
 w5.fgdPrompt = YELLOW
 w5.bgdPrompt = RED
 w5Title$ = " W A R N I N G "
 w5Text$(1) = ""
 w5Text$(2) = "       This action will slew the telescope. "
 w5Text$(3) = " The initial zenith angle will be " + Zenith$
+ " deg. "
 w5Text$(4) = " Any key other than F1 exits this procedure. "
 w5Text$(5) = " The ESC key will halt telescope movement at "
 w5Text$(6) = " any time."
 w5Text$(7) = ""
 w5Prompt$ = "Press the F1 key to slew"
'
'The Change telescope status window
 w6.action = 1
 w6.edgeline = 1
 w6.row = 5
 w6.col = 3
 w6.fgdEdge = CYAN + BRIGHT
 w6.bgdEdge = BLUE
 w6.fgdBody = BRIGHT + WHITE
 w6.bgdBody = BLUE
 w6.fgdHighlight = 0
 w6.bgdHighlight = 0
 w6.fgdTitle = WHITE + BRIGHT
 w6.bgdTitle = BLUE
 w6.fgdPrompt = YELLOW
 w6.bgdPrompt = BLUE
 w6Title$ = " Change of Telescope Status "
 w6Text$(1) = ""
 w6Text$(2) = " F1:  Edit the current list "
 w6Text$(3) = " F2:  Choose a star from the current list "
 w6Text$(4) = " F3:  Change to a new data file "
 w6Text$(5) = " F4:  Change atmospheric parameters "
 w6Text$(6) = " F5:  Park the telescope "
 w6Text$(7) = " ESC: Exit the program and halt the telescope
"
 w6Text$(8) = " Any other key aborts the change "
 w6Text$(9) = ""
 w6Prompt$ = "Choose an option"
'
'The RA/DEC format error window
 w7.action = 1
 w7.edgeline = 1
 w7.row = 5
 w7.col = 5
 w7.fgdEdge = CYAN + BRIGHT
 w7.bgdEdge = RED
 w7.fgdBody = BRIGHT + WHITE
```

```
 w7.bgdBody = RED
 w7.fgdHighlight = 0
 w7.bgdHighlight = 0
 w7.fgdTitle = CYAN
 w7.bgdTitle = RED
 w7.fgdPrompt = YELLOW
 w7.bgdPrompt = RED
 w7Title$ = ""
 w7Text$(1) = " Error - RA or DEC is formatted incorrectly for
this star "
 w7Prompt$ = "Press any key to continue"
'
'The slew error window
 w8.action = 1
 w8.edgeline = 1
 w8.row = 5
 w8.col = 10
 w8.fgdEdge = CYAN + BRIGHT
 w8.bgdEdge = RED
 w8.fgdBody = BRIGHT + WHITE
 w8.bgdBody = RED
 w8.fgdHighlight = 0
 w8.bgdHighlight = 0
 w8.fgdTitle = CYAN + BLINK
 w8.bgdTitle = RED
 w8.fgdPrompt = YELLOW
 w8.bgdPrompt = RED
 w8Title$ = " NOTICE "
 w8Text$(1) = "      This star is not in view or has a zenith
angle "
 w8Text$(2)  =  "  greater  than  75  degrees.   Please  choose
another star."
 w8Prompt$ = "Press any key to continue"
'
'The zenith angle warning window
 w9.action = 0
 w9.edgeline = 1
 w9.row = 7
 w9.col = 57
 w9.fgdEdge = CYAN + BRIGHT
 w9.bgdEdge = YELLOW
 w9.fgdBody = BRIGHT + WHITE
 w9.bgdBody = YELLOW
 w9.fgdHighlight = 0
 w9.bgdHighlight = 0
 w9.fgdTitle = CYAN + BLINK
 w9.bgdTitle = YELLOW
 w9.fgdPrompt = YELLOW
 w9.bgdPrompt = YELLOW
 w9Title$ = " NOTICE "
 w9Text$(1) = " The zenith "
```

```
 w9Text$(2) = " angle is"
 w9Text$(3) = STR$(Zenith%) + " degrees "
 w9Prompt$ = ""
'
'The zenith angle halt movement window
 w10.action = 1
 w10.edgeline = 1
 w10.row = 5
 w10.col = 5
 w10.fgdEdge = CYAN + BRIGHT
 w10.bgdEdge = RED
 w10.fgdBody = BRIGHT + WHITE
 w10.bgdBody = RED
 w10.fgdHighlight = 0
 w10.bgdHighlight = 0
 w10.fgdTitle = CYAN + BLINK
 w10.bgdTitle = RED
 w10.fgdPrompt = YELLOW
 w10.bgdPrompt = RED
 w10Title$ = " NOTICE "
 w10Text$(1) = " Telescope movement has been halted. "
 w10Text$(2) = " The zenith angle has exceeded 75 degrees. "
 w10Prompt$ = "Press any key to continue"
'
'The Status help window
 w11.action = 1
 w11.edgeline = 1
 w11.row = 5
 w11.col = 3
 w11.fgdEdge = CYAN + BRIGHT
 w11.bgdEdge = BLUE
 w11.fgdBody = BRIGHT + WHITE
 w11.bgdBody = BLUE
 w11.fgdHighlight = 0
 w11.bgdHighlight = 0
 w11.fgdTitle = WHITE + BRIGHT
 w11.bgdTitle = BLUE
 w11.fgdPrompt = YELLOW
 w11.bgdPrompt = BLUE
 w11Title$ = " Telescope Status Help"
 w11Text$(1) = "  Slow step:                          Fast step:
"
 w11Text$(2) = "                                     "
 w11Text$(3) = "  up arrow    = Increase DEC     8 = Increase
DEC "
 w11Text$(4) = "  down arrow  = Decrease DEC     2 = Decrease
DEC "
 w11Text$(5) = "  left arrow  = Increase RA      4 = Increase
RA "
 w11Text$(6) = "  right arrow = Decrease RA      6 = Decrease
RA "
```

```
 w11Text$(7) = "                                          "
 w11Text$(8) = "  Slew: "
 w11Text$(9) = "                                          "
 w11Text$(10) = "  1 = Increase DEC                       "
 w11Text$(11) = "  3 = Decrease DEC                      "
 w11Text$(12) = "  7 = Increase RA                       "
 w11Text$(13) = "  9 = Decrease RA                       "
 w11Prompt$ = " Press any key to exit help "
'
'The CTS-10 error window
 w12.action = 1
 w12.edgeline = 1
 w12.row = 5
 w12.col = 5
 w12.fgdEdge = CYAN + BRIGHT
 w12.bgdEdge = RED
 w12.fgdBody = BRIGHT + WHITE
 w12.bgdBody = RED
 w12.fgdHighlight = 0
 w12.bgdHighlight = 0
 w12.fgdTitle = CYAN + BLINK
 w12.bgdTitle = RED
 w12.fgdPrompt = YELLOW
 w12.bgdPrompt = RED
 w12Title$ = "Error"
 w12Text$(1) = " The CTS-10 WWV Time Syncronization board is
not "
 w12Text$(2) = " responding.  Take one of the following
actions: "
 w12Text$(3) = "  "
 w12Text$(4) = "     a.  Check the CTS-10 onboard battery "
 w12Text$(5) = "     b.  Manually update the system time to
UTC "
 w12Prompt$ = "Press any key to exit"
'
'Initialize the display
 SCREEN 0, , 0, 0
 WIDTH 80
 InitialDisplay
'
'************************************
'**  Temporary Debugging Statement **
 DO
 LOOP UNTIL INKEY$ <> ""
'************************************
'
'Initialize the rest of the system (describe)
'************************************************************
*************
'**
```

```
'**                        More Work Needed
        **
'POKE (address of external boards), 0
        **
'POKE (address of Key paddle buffer), 0
        **
'************************************************************
*************
'
'Check the CTS-10 WWV Time Synch board for response
 year% = VAL(RIGHT$(DATE$, 4))
 IF year% < 1992 THEN windows w12, w12Text$(), w12Title$,
w12Prompt$
 SYSTEM
'
'Perform a one-time calculation of the Equation of Equinox
correction to
'sidereal time
 ON ERROR GOTO EQofEQError:
        EquationEquinoxes EQofEQ(), EQofEQUIN!
 ON ERROR GOTO 0
'
'Determine current time and date information for use in
DayLightSaving,
'SiderealTimeCalc SUBRoutines and Status DO LOOP, below
 min% = VAL(MID$(TIME$, 4, 2))
 hour% = VAL(LEFT$(TIME$, 2))
 day% = VAL(MID$(DATE$, 4, 2))
 month% = VAL(LEFT$(DATE$, 2))
 year% = VAL(RIGHT$(DATE$, 4))
'
'Check whether Daylight Savings time applies to local time
 DayLightSaving DayLightSavings%, hour%, day%, month%, year%
'
 ChooseDecision% = TRUE
 AtmosChange% = FALSE
 DataEntry% = TRUE
 RADecError% = FALSE
 firstloop% = TRUE
 WindowOn% = FALSE
 Simulation% = FALSE
 DatafileOld$ = "STARFILE.DAT"
 Pressure! = 633!  'mmHg @ 5000 ft altitude default
 Temperature! = 10! 'C @ 5000 ft altitude default
 Wavelength! = .5  'microns @ 5000 ft altitude default
'
'Allow the user to change the default atmospheric conditions
used to calculate refraction
 Atmospheric Pressure!, Temperature!, Wavelength!
'
'Perform sidereal time calculations
```

```
 SiderealTimeCalc SiderealTime$, min%, hour%, day%, month%,
year%, EQofEQUIN!, GMST0hrSEC#, JulianDate#, dayFraction#,
DaysPassed%
'
'Align telescope in software

 TeleAlign GMST0hrSEC#, GHAaries#, TeleRA#, TeleDEC#
'
 DO
        'Perform sidereal time calculations
        SiderealTimeCalc SiderealTime$, min%, hour%, day%,
month%,   year%,   EQofEQUIN!,   GMST0hrSEC#,   JulianDate#,
dayFraction#, DaysPassed%
        '
        IF DataEntry% = TRUE THEN
                DataEntry% = FALSE
                'Accept star data entry from disk or by hand
input
                COLOR WHITE, BLUE, BLUE
                FOR r% = 14 TO 21
                        LOCATE r%, 6, 0
                        PRINT SPACE$(68)
                NEXT r%
                EditDecision% = FALSE
                NewFile% = TRUE
                DO
                        COLOR WHITE, BLUE, BLUE
                        LOCATE 17, 10, 0
                        PRINT SPACE$(45)
                        LOCATE 17, 10, 0
                        PRINT "Enter star data by:  (F1) Disk"
                        LOCATE 18, 31, 0
                        PRINT "(F2) Keyboard"
                        kee% = KeyCode%(Character$)
                        SELECT CASE kee%
                                CASE 15104 'F1
                                        BadPathType% = FALSE
                                        NewFile% = FALSE
                                        FOR i% = 1 TO 2
                                                LOCATE  16  +
i%, 25, 0
                                                         PRINT
SPACE$(34)
                                        NEXT i%
                                        LOCATE 17, 47, 1
                                        PRINT DatafileOld$
                                        LOCATE 17, 10, 1
                                        INPUT  "Please  enter
the path and file name: ", Datafile$
                                        IF LEN(Datafile$) = 0
THEN
```

```
Datafile$ = DatafileOld$
ELSE
DatafileOld$ = Datafile$
END IF
IF LEN(Datafile$) > 0 THEN
ON ERROR GOTO BadPath:
OPEN Datafile$ FOR INPUT AS #1
ON ERROR GOTO 0
IF BadPathType% = FALSE THEN
i% = 1
DO WHILE NOT EOF(1)
  INPUT #1, Star$(i%), RA$(i%), DEC$(i%), EPOCH$(i%)
  i% = i% + 1
LOOP
IFi% < 100 THEN
  FOR h% = i% TO 100
          Star$(h%) = ""
          RA$(h%) = ""
          DEC$(h%) = ""
          EPOCH$(h%) = ""
  NEXT h%
END IF
RESET
LOCATE 17, 11, 0
PRINT SPACE$(63)
LOCATE 17, 10, 0
PRINT "Do you wish to edit the file"
LOCATE 18, 10, 0
```

```
                                                              PRINT
"prior to proceeding? (Y or N) "
                                                              COLOR
WHITE, BLUE, BLUE
                                                              LOCATE
18, 40, 0
                                                              PRINT
"N"
                                                              LOCATE
18, 40, 1
                                                              kee%
= KeyCode%(Character$)
                                                              IF
UCASE$(Character$) = "Y" THEN EditDecision% = TRUE
                                                              Stops%
= TRUE
                                                        ELSE
                                                              'no
action, return to beginning of loop
                                                        END IF
                                                ELSE
                                                        'no  action,
return to beginning of loop
                                                END IF
                                        CASE 15360 ' F2
                                                EditDecision% = TRUE
                                                NewFile% = TRUE
                                                Stops% = TRUE
                                        CASE ELSE
                                                Stops% = FALSE
                                END SELECT
                        LOOP UNTIL Stops% = TRUE
                END IF
                '
                IF EditDecision% = TRUE THEN
                        EditDecision% = FALSE
                        'Change the screen to display readiness for
star data entry
                        StarDisplay
                        'Accept star data entry
                        StarDataEntry   Star$(),   RA$(),   DEC$(),
EPOCH$(), NewFile%, w1, w1Text$(), w1Title$, w1Prompt$
                        '
                        'Save data to file
                        IF NewFile% = TRUE THEN
                                NewFile% = FALSE
                                COLOR WHITE, BLUE, BLUE
                                FOR r% = 14 TO 21
                                        LOCATE r%, 6, 0
                                        PRINT SPACE$(68)
                                NEXT r%
```

```
                        COLOR BLUE, BLACK, BLUE
                        LOCATE 23, 5, 0
                        PRINT STRING$(30, 177)
                        Datafile$ = "StarFile.DAT"
                        COLOR WHITE, BLUE, BLUE
                        LOCATE 15, 37, 0
                        PRINT Datafile$
                        LOCATE 15, 10, 0
                        INPUT "Please enter the filename: ",
DataFile2$
                        LOCATE 16, 10, 0
                        COLOR CYAN + BLINK, BLUE, BLUE
                        PRINT "SAVING"
                        IF LEN(DataFile2$) > 0 THEN Datafile$
= DataFile2$
                        ON ERROR GOTO NewFileError:
                                OPEN Datafile$ FOR OUTPUT AS
#1
                        ON ERROR GOTO 0
                        FOR i% = 1 TO UBOUND(Star$)
                                WRITE #1, Star$(i%), RA$(i%),
DEC$(i%), EPOCH$(i%)
                        NEXT i%
                        RESET
                ELSE
                        'If an old file, store to disk
                        COLOR WHITE, BLUE, BLUE
                        FOR r% = 14 TO 21
                                LOCATE r%, 6, 0
                                PRINT SPACE$(68)
                        NEXT r%
                        COLOR BLUE, BLACK, BLUE
                        LOCATE 23, 5, 0
                        PRINT STRING$(30, 177)
                        COLOR WHITE, BLUE, BLUE
                        LOCATE 15, 37, 0
                        PRINT Datafile$
                        LOCATE 15, 10, 0
                        INPUT "Please enter the filename: ",
DataFile2$
                        LOCATE 16, 10, 0
                        COLOR CYAN + BLINK, BLUE, BLUE
                        PRINT "SAVING"
                        IF LEN(DataFile2$) > 0 THEN Datafile$
= DataFile2$
                        ON ERROR GOTO OldFileError:
                                OPEN Datafile$ FOR OUTPUT AS
#1
                        ON ERROR GOTO 0
                        IF FileError% = FALSE THEN
                                FOR i% = 1 TO 100
```

```
                                        WRITE #1, Star$(i%),
RA$(i%), DEC$(i%), EPOCH$(i%)
                                NEXT i%
                        END IF
                        RESET
                        FOR r% = 14 TO 21
                                LOCATE r%, 6, 0
                                PRINT SPACE$(68)
                        NEXT r%
                END IF
        END IF
        '
        'Allow the user to update the atmospheric parameters
on which refraction is calculated
        IF AtmosChange% = TRUE THEN
                AtmosChange% = FALSE
                Atmospheric    Pressure!,    Temperature!,
Wavelength!
        END IF
        '
        'Allow the user to choose the active star on which the
telescope will track
        IF ChooseDecision% = TRUE THEN
                ChooseDecision% = FALSE
                ChooseStar Star$(), RA$(), DEC$(), EPOCH$(),
NewFile%, w1, w1Text$(), w1Title$, w1Prompt$, currentStar%
                ActiveStar$ = Star$(currentStar%)
                'Calculate Precession and Nutation effects for
the day
                Precession    RA$(),    DEC$(),    EPOCH$(),
currentStar%, JulianDate#, EpochError%, RAofStar#, DECofStar#,
RADecError%
                '
                'RA  and  DEC  displayed  are  corrected  for
precession only
                RAofStarDisplay# = RAofStar#
                DECofStarDisplay# = DECofStar#
                '
                Nutation RAofStar#, DECofStar#, JulianDate#,
N#(), S1#(), S2#()
                '
                 Aberration    RAofStar#,    DECofStar#,
RAAberrationCor#, DecAberrationCor#, dayFraction#, DaysPassed%
                RAofStar# = RAofStar# + RAAberrationCor#
                DECofStar# = DECofStar# + DecAberrationCor#
                '
                 Refraction    Pressure!,    Temperature!,
Wavelength!, GMST0hrSEC#, HourAngle#, GHAaries#, RAofStar#,
DECofStar#, RARefractCor#, DecRefractCor#, RAofMIRA#
                RAofStar# = RAofStar# + RARefractCor#
                DECofStar# = DECofStar# + DecRefractCor#
```

```
                RARefractCorOld# = RARefractCor#
                DecRefractCorOld# = DecRefractCor#
                DO WHILE RAofStar# >= (2# * pi)
                        RAofStar# = RAofStar# - (2# * pi)
                LOOP
                DO WHILE RAofStar# < 0
                        RAofStar# = RAofStar# + (2# * pi)
                LOOP
                DO WHILE DECofStar# > (pi / 2#)
                        DECofStar# = pi - DECofStar#
                LOOP
                DO WHILE DECofStar# < -(pi / 2#)
                        DECofStar# = -(pi + DECofStar#)
                LOOP
                TeleAngles  Zenith%,  Azimuth%,  GHAaries#,
RAofStar#, DECofStar#, CelCoord#(), DCM#(), TerraRec#()
                Zenith$ = STR$(Zenith%)
                w5Text$(3) = " The initial zenith angle will
be " + Zenith$ + " deg. "
                windows w5, w5Text$(), w5Title$, w5Prompt$
        ELSE
                w5.returnCode% = 15104' F1
        END IF
        '
        'If the user enters F10 vice F1, simulation of the
control process occurs
        IF w5.returnCode = 17408 THEN ' F10
                w5.returnCode = 15104'  F1
                Simulation% = TRUE
        END IF
        SELECT CASE w5.returnCode
                CASE 15104' F1
                        WindowsPop
                        TimeSec% = VAL(RIGHT$(TIME$, 2))
                        '
                        'Display telescope status screen
                         TeleStatusDisplay   ActiveStar$,
Simulation%
                        '
                        Quit% = FALSE
                        FirstPass% = TRUE
                        DO
                                'Calculate and display local,
sidereal and UTC times
                                'and  calculate  Refraction
corrections
                                IF  TimeSec%  =  59  AND
VAL(RIGHT$(TIME$, 2)) = 0 THEN
                                       ShowTime min%, hour%,
day%, month%, year%, DayLightSavings%, SiderealTime$
```

```
                                        TimeSec%    =
VAL(RIGHT$(TIME$, 2))
                                    RefractionPressure!,
Temperature!, Wavelength!, GMST0hrSEC#, HourAngle#, GHAaries#,
RAofStar#,   DECofStar#,   RARefractCor#,   DecRefractCor#,
RAofMIRA#
                                    RAofStar# = RAofStar#
- RARefractCorOld#
                                       DECofStar#    =
DECofStar# - DecRefractCorOld#
                                    RAofStar# = RAofStar#
+ RARefractCor#
                                       DECofStar#    =
DECofStar# + DecRefractCor#
                                    RARefractCorOld#   =
RARefractCor#
                                    DecRefractCorOld#  =
DecRefractCor#
                                ELSEIF   TimeSec%    <
VAL(RIGHT$(TIME$, 2)) THEN
                                    ShowTime min%, hour%,
day%, month%, year%, DayLightSavings%, SiderealTime$
                                        TimeSec%    =
VAL(RIGHT$(TIME$, 2))
                                    RefractionPressure!,
Temperature!, Wavelength!, GMST0hrSEC#, HourAngle#, GHAaries#,
RAofStar#,   DECofStar#,   RARefractCor#,   DecRefractCor#,
RAofMIRA#
                                    RAofStar# = RAofStar#
- RARefractCorOld#
                                       DECofStar#    =
DECofStar# - DecRefractCorOld#
                                    RAofStar# = RAofStar#
+ RARefractCor#
                                       DECofStar#    =
DECofStar# + DecRefractCor#
                                    RARefractCorOld#   =
RARefractCor#
                                    DecRefractCorOld#  =
DecRefractCor#
                            END IF
                            DO WHILE RAofStar# >= (2# *
pi)
                                    RAofStar# = RAofStar#
- (2# * pi)
                            LOOP
                            DO WHILE RAofStar# < 0
                                    RAofStar# = RAofStar#
+ (2# * pi)
                            LOOP
```

```
                    DO WHILE DECofStar# > (pi /
2#)
                            DECofStar# = pi -
DECofStar#
                    LOOP
                    DO WHILE DECofStar# < -(pi /
2#)
                            DECofStar# = -(pi +
DECofStar#)
                    LOOP
                    '
                    'Display current RA, DEC and
Hour Angle of the star
                    COLOR WHITE, BLUE, BLUE
                        RAofStarInHours# =
RAofStarDisplay# * 24# / (2# * pi)
                                RAHours% =
INT(RAofStarInHours#)
                              RAMinutes% =
INT((RAofStarInHours# - CDBL(RAHours%)) * 60#)
                              RASeconds! =
((RAofStarInHours# - CDBL(RAHours%)) * 60# - CDBL(RAMinutes%))
* 60#
                    LOCATE 15, 10, 0
                                       PRINT
RTRIM$(LTRIM$(STR$(RAHours%))); ":";
RTRIM$(LTRIM$(STR$(RAMinutes%))); ":";
RTRIM$(LTRIM$(STR$(INT(RASeconds! * 100) / 100))); SPACE$(5)
                    DECinDeg# = DECofStarDisplay#
* 360# / (2# * pi)
                    DECinDeg! = CSNG(DECinDeg#)
                    DECDegrees% = INT(DECinDeg!)
                    DECMinutes% = INT((DECinDeg#
- CDBL(DECDegrees%)) * 60#)
                             DECSeconds! =
CSNG(((DECinDeg# - CDBL(DECDegrees%)) * 60#) -
CDBL(DECMinutes%)) * 60#)
                    LOCATE 18, 10, 0
                                       PRINT
RTRIM$(LTRIM$(STR$(DECDegrees%))); CHR$(248);
RTRIM$(LTRIM$(STR$(DECMinutes%))); "'";
RTRIM$(LTRIM$(STR$(INT(DECSeconds! * 100) / 100))); "''";
SPACE$(5)
                    '
                    'Update the star's hour angle
and display
                    HourAngle# = RAofMIRA# -
RAofStar#   'in radians
                    DO WHILE HourAngle# >= (2# *
pi)
```

```
                                        HourAngle#    =
HourAngle# - (2# * pi)
                                LOOP
                                DO WHILE HourAngle# < 0
                                        HourAngle#    =
HourAngle# + (2# * pi)
                                LOOP
                                HourAngleInDeg# = HourAngle#
* 360# / (2# * pi)
                                IF HourAngleInDeg# > 180 THEN
                                     HourAngleInDeg#    =
HourAngleInDeg# - 360#
                                END IF
                                     HourAngleInHr#    =
HourAngleInDeg# / 15#'Conversion of degrees to hours of time
                                    HourAngleHours$    =
STR$(FIX(HourAngleInHr#))
                                IF VAL(HourAngleHours$) = 0
AND HourAngleInDeg# < 0 THEN HourAngleHours$ = "-0"
                                     HourAngleMin#    =
ABS((HourAngleInHr# - CDBL(FIX(HourAngleInHr#)))) * 60#
                                   HourAngleMinutes$    =
STR$(FIX(HourAngleMin#))
                                IF LEN(HourAngleMinutes$) = 1
THEN HourAngleMinutes$ = "0" + HourAngleMinutes$
                               HourAngleSec# = (HourAngleMin#
- CDBL(FIX(HourAngleMin#))) * 60#
                                   HourAngleSecMod%    =
FIX(HourAngleSec# * 100#)
                                   HourAngleSeconds$    =
STR$(ABS(CSNG(HourAngleSecMod%) / 100!))
                                LOCATE 21, 10, 0
                                                   PRINT
LTRIM$(RTRIM$(HourAngleHours$));  ":";
LTRIM$(RTRIM$(HourAngleMinutes$));  ":";
LTRIM$(HourAngleSeconds$); SPACE$(10)
                                '
                                'Calculate and display Zenith
Angle and Azimuth Angle on status screen
                                TeleAngles Zenith%, Azimuth%,
GHAaries#, RAofStar#, DECofStar#, CelCoord#(), DCM#(),
TerraRec#()
                                LOCATE 15, 37, 0
                                PRINT Zenith%;
                                LOCATE 15, POS(0) - 1, 0
                                PRINT CHR$(248); SPACE$(5)
                                LOCATE 18, 36, 0
                                PRINT Azimuth%;
                                LOCATE 18, POS(0) - 1, 0
                                PRINT CHR$(248); SPACE$(5)
                                IF RADecError% = TRUE THEN
```

```
                                        RADecError% = FALSE
                                        kee% = ENTER
                                       windows w7, w7Text$(),
w7Title$, w7Prompt$
                                        WindowsPop
                                ELSEIF  Zenith%  >=  75  AND
firstloop% = TRUE THEN
                                        firstloop% = FALSE
                                       windows w8, w8Text$(),
w8Title$, w8Prompt$
                                        WindowsPop
                                        Quit% = TRUE
                                ELSEIF  Zenith%  >=  75  AND
firstloop% = FALSE THEN
                                            windows   w10,
w10Text$(), w10Title$, w10Prompt$
                                        WindowsPop
                                        Quit% = TRUE
                                ELSE
                                        IF Zenith% > 65 AND
WindowOn% = FALSE THEN
                                                WindowOn%  =
TRUE
                                                ZenithOld% =
Zenith%
                                                w9Text$(3) =
STR$(Zenith%) + " degrees "
                                                windows  w9,
w9Text$(), w9Title$, w9Prompt$
                                        END IF
                                        IF Zenith% > 65 AND
WindowOn% = TRUE THEN
                                                IF Zenith% <>
ZenithOld% THEN
ZenithOld% = Zenith%
WindowsPop
w9Text$(3) = STR$(Zenith%) + " degrees "
windows w9, w9Text$(), w9Title$, w9Prompt$
                                                END IF
                                        END IF
                                         IF   Zenith%    <>
ZenithOld% AND WindowOn% = TRUE THEN
                                                ZenithOld% =
Zenith%
                                                WindowsPop
                                                WindowOn%  =
FALSE
```

```
END IF
'
'Clear the motor state display boxes
COLOR WHITE, BLUE, BLUE
FOR i% = 1 TO 2
        LOCATE 3, 10 * i%, 0
                PRINT SPACE$(1)
NEXT i%
FOR i% = 1 TO 2
        LOCATE 3, (10 * i%) + 50, 0
                PRINT SPACE$(1)
NEXT i%
'
'Read in Telescope position data from encoders and operate on them
'to define TeleRA# and TeleDEC# in RADIANS
'*******************************
'**   Read in encoder data   **
'*******************************
RAdiff# = TeleRA# - RAofStar#
DECdiff# = TeleRA# - DECofStar#
IF FirstPass% = TRUE THEN
        IF ABS(RAdiff#) > .0175 OR ABS(DECdiff#) > .0175 THEN
                SlewCommands Simulation%, TeleRA#, TeleDEC#, RAofStar#, DECofStar#
        END IF
        RAdiff# = TeleRA# - RAofStar#
        DECdiff# = TeleRA# - DECofStar#
        IF ABS(RAdiff#) < .0175 OR ABS(DECdiff#) < .0175 THEN
                SetCommands Simulation%, TeleRA#, TeleDEC#, RAofStar#, DECofStar#
        END IF
```

```
                                                       FirstPass% =
FALSE
                                               END IF
                                               TrackCommands
                                                          kee%     =
KeyCodeNoWait%(Character$)
                                       END IF
                                       SELECT CASE kee%
                                               CASE ENTER
                                                       Quit% = TRUE
                                               CASE ESCAPE
                                                       windows  w1,
w1Text$(), w1Title$, w1Prompt$
                                                       WindowsPop
                                                                 IF
w1.returnCode = 121 OR w1.returnCode = 89 THEN Abort% = TRUE
                                                       IF Abort% =
TRUE THEN
'**********************************
                                                                '**
System Shutdown Procedures  **
                                                               '**
       go here!!           **
'**********************************
                                                               CLS
                                                             SYSTEM
                                                       END IF
                                               CASE F1
                                                       windows w11,
w11Text$(), w11Title$, w11Prompt$
                                                       WindowsPop
                                               CASE LEFTARROW
                                                       'Change   RA
step multiplier to SLOW speed, INCREASE right ascension
                                                       '
                                                     'POKE (address
of RA step controller), X
                                                       'Updatemotor
display box
                                                       COLOR   RED,
BLUE, BLUE
                                                       LOCATE 3, 20,
0
                                                              PRINT
CHR$(222)
                                               CASE RIGHTARROW
                                                       'Change   RA
step multiplier to SLOW speed, DECREASE right ascension
                                                       '
```

```
                                                        'POKE (address
of RA step controller), X
                                                          'Updatemotor
display box
                                                          COLOR   RED,
BLUE, BLUE
                                                          LOCATE 3, 20,
0
                                                                 PRINT
CHR$(222)
                                                  CASE NUMLEFTARROW
                                                          'Change   RA
step multiplier to FAST speed, INCREASE right ascension
                                                          '
                                                        'POKE (address
of RA step controller), X
                                                          'Updatemotor
display box
                                                          COLOR   RED,
BLUE, BLUE
                                                          LOCATE 3, 20,
0
                                                                 PRINT
CHR$(222)
                                                  CASE NUMRIGHTARROW
                                                          'Change   RA
step multiplier to FAST speed, DECREASE right ascension
                                                          '
                                                        'POKE (address
of RA step controller), X
                                                          'Updatemotor
display box
                                                          COLOR   RED,
BLUE, BLUE
                                                          LOCATE 3, 20,
0
                                                                 PRINT
CHR$(222)
                                                  CASE UPARROW
                                                          'Change  DEC
step multiplier to SLOW speed, INCREASE declination
                                                          '
                                                        'POKE (address
of DEC step controller), X
                                                          'Updatemotor
display box
                                                          COLOR   RED,
BLUE, BLUE
                                                          LOCATE 3, 70,
0
```

```
                                                        PRINT
CHR$(222)
                                          CASE DOWNARROW
                                                'Change  DEC
step multiplier to SLOW speed, DECREASE declination
                                                '
                                               'POKE (address
of DEC step controller), X
                                                'Updatemotor
display box
                                                COLOR   RED,
BLUE, BLUE
                                                LOCATE 3, 70,
0
                                                        PRINT
CHR$(222)
                                          CASE NUMUPARROW
                                                'Change  DEC
step multiplier to FAST speed, INCREASE declination
                                                '
                                               'POKE (address
of DEC step controller), X
                                                'Updatemotor
display box
                                                COLOR   RED,
BLUE, BLUE
                                                LOCATE 3, 70,
0
                                                        PRINT
CHR$(222)
                                          CASE NUMDOWNARROW
                                                'Change  DEC
step multiplier to FAST speed, DECREASE declination
                                                '
                                               'POKE (address
of DEC step controller), X
                                                'Updatemotor
display box
                                                COLOR   RED,
BLUE, BLUE
                                                LOCATE 3, 70,
0
                                                        PRINT
CHR$(222)
                                          CASE seven
                                               'Manually slew
to INCREASE right ascension
                                                '
                                                'Updatemotor
display box
```

```
COLOR RED,
BLUE, BLUE
LOCATE 3, 10,
0
PRINT
CHR$(222)
'
'POKE (address
of RA slew controller), X
CASE nine
'Manually slew
to DECREASE right ascension
'
'Updatemotor
display box
COLOR RED,
BLUE, BLUE
LOCATE 3, 10,
0
PRINT
CHR$(222)
'POKE (address
of RA slew controller), X
CASE one
'Manually slew
to INCREASE declination
'
'Updatemotor
display box
COLOR RED,
BLUE, BLUE
LOCATE 3, 60,
0
PRINT
CHR$(222)
'POKE (address
of DEC slew controller), X
CASE three
'Manually slew
to DECREASE declination
'
'Updatemotor
display box
COLOR RED,
BLUE, BLUE
LOCATE 3, 60,
0
PRINT
CHR$(222)
'POKE (address
of DEC slew controller), X
```

```
                                                CASE ELSE
                                                        'Turn   slew
motors OFF once no key is pressed
                                                        '
                                                        'POKE (address
of RA slew controller), X
                                                        'POKE (address
of DEC slew controller), X
                                        END SELECT
                                LOOP UNTIL Quit% = TRUE
                                IF WindowOn% = TRUE THEN
                                        WindowOn% = FALSE
                                        WindowsPop
                                END IF
                                Quit% = FALSE
                                windows  w6,  w6Text$(),  w6Title$,
w6Prompt$
                                WindowsPop
                                SELECT CASE w6.returnCode%
                                        CASE 15104 ' F1
                                                EditDecision% = TRUE
                                               ChooseDecision% = TRUE
                                                firstloop% = TRUE
                                        CASE 15360 ' F2
                                               ChooseDecision% = TRUE
                                                firstloop% = TRUE
                                        CASE 15616 ' F3
                                                DataEntry% = TRUE
                                               ChooseDecision% = TRUE
                                                firstloop% = TRUE
                                        CASE 15872 ' F4
                                                AtmosChange% = TRUE
                                               ChooseDecision% = TRUE
                                                firstloop% = TRUE
                                        CASE 16128 ' F5

'*****************************
                                                '**      Park   the
telescope    **

'*****************************
                                        CASE ESCAPE
                                               windows w1, w1Text$(),
w1Title$, w1Prompt$
                                                WindowsPop
                                                IF w1.returnCode = 121
OR w1.returnCode = 89 THEN Abort% = TRUE
                                                IF Abort% = TRUE THEN

'*************************************
```

```
                                                          '**   System
Shutdown Procedures  **
                                                          '**
go here!!           **
'***********************************
                                                         CLS
                                                         SYSTEM
                                                 END IF
                                         CASE ELSE
                                 END SELECT
                                 '
                         CASE ELSE
                                 WindowsPop
                                 windows  w6,  w6Text$(),  w6Title$,
w6Prompt$
                                 WindowsPop
                                 SELECT CASE w6.returnCode%
                                         CASE 15104 ' F1
                                                 EditDecision% = TRUE
                                                ChooseDecision% = TRUE
                                         CASE 15360 ' F2
                                                ChooseDecision% = TRUE
                                         CASE 15616 ' F3
                                                 DataEntry% = TRUE
                                                ChooseDecision% = TRUE
                                         CASE 15872 ' F4
                                                 AtmosChange% = TRUE
                                                ChooseDecision% = TRUE
                                         CASE 16128 ' F5
'*****************************
                                                 '**      Park   the
telescope    **
'*****************************
                                         CASE ESCAPE
                                                windows w1, w1Text$(),
w1Title$, w1Prompt$
                                                 WindowsPop
                                                 IF w1.returnCode = 121
OR w1.returnCode = 89 THEN Abort% = TRUE
                                                 IF Abort% = TRUE THEN
'***********************************
                                                          '**   System
Shutdown Procedures  **
                                                          '**
go here!!           **
'***********************************
```

```
                                        CLS
                                        SYSTEM
                                END IF
                        CASE ELSE
                END SELECT
        END SELECT
        '
 LOOP
'
'Error Handling and time updates
 GOTO EndLabel:
'
NewFileError:
        RESET
        BEEP
        windows w2, w2Text$(), w2Title$, w2Prompt$
        WindowsPop
        FileError% = TRUE
        RESUME NEXT
'
OldFileError:
        RESET
        BEEP
        windows w2, w2Text$(), w2Title$, w2Prompt$
        WindowsPop
        FileError% = TRUE
        RESUME NEXT
'
BadPath:
        RESET
        BadPathType% = TRUE
        BEEP
        windows w3, w3Text$(), w3Title$, w3Prompt$
        WindowsPop
        RESUME NEXT
'
EQofEQError:
        BEEP
        windows w4, w4Text$(), w4Title$, w4Prompt$
        WindowsPop
        RESUME
'
EndLabel:
END

'****************************************
'**     Name:       Aberration         **
'**     Type:       Subroutine         **
'**     Module:     MIRACTRL.BAS       **
'**     Language:   Quickbasic 4.50    **
'****************************************
```

```
'
'Calculates the aberration correction to celestial
coordinates.
'
'Example of use: Aberration RAofStar#, DECofStar#,
RAAberrationCor#, DecAberrationCor#, dayFraction#, DaysPassed%
'
SUB Aberration (RAofStar#, DECofStar#, RAAberrationCor#,
DecAberrationCor#, dayFraction#, DaysPassed%)
'
'Calculate the mean longitude of the Sun, L#
 d# = dayFraction# + CDBL(DaysPassed%)
 L# = 298.926 + .985647 * d#    '  in degrees
 L# = L# * (2# * pi) / 360#     '  Convert to radians
'
'Define the constant of aberration, k#
 k# = 20.496                    '  in arc seconds
 k# = k# * (2# * pi) / (360# * 3600#)'  convert to radians
'
'Define the obliquity of the ecliptic
 epsilon# = 23.440332# - .00000036# * d#  'degrees
 epsilon# = epsilon# * (2# * pi) / 360#   'radians
'
'Calculate the corrections to RA and Declination due to
aberration
 RAAberrationCor# = (-k# * SIN(L#) * SIN(RAofStar#) - k# *
COS(L#) * COS(epsilon#) * COS(RAofStar#)) / COS(DECofStar#)
 DecAberrationCor# = -k# * SIN(L#) * COS(RAofStar#) *
SIN(DECofStar#) + k# * COS(L#) * (COS(epsilon#) *
SIN(RAofStar#) * SIN(DECofStar#) - SIN(epsilon#) *
COS(DECofStar#))
'
END SUB

'****************************************
'**     Name:       Atmospheric       **
'**     Type:       Subroutine        **
'**     Module:     MIRACTRL.BAS      **
'**     Language:   Quickbasic 4.50   **
'****************************************
'
'Allows user entry of atmospheric temperature, pressure and
the center
'wavelength of light for an observation.  These data are used
in refraction
'calculations.
'
'Example of use: Atmospheric Pressure!, Temperature!,
Wavelength!
'
SUB Atmospheric (Pressure!, Temperature!, Wavelength!)
```

```
'
'Clear the body of the status box
 COLOR WHITE, BLUE, BLUE
 FOR r% = 14 TO 21
    LOCATE r%, 6, 0
    PRINT SPACE$(68);
 NEXT r%
'
 LOCATE 15, 55, 0
 PRINT Temperature!
 LOCATE 15, 10, 0
 INPUT "Please enter the atmospheric temperature (C): ",
NewTemp$
 IF LEN(NewTemp$) > 0 THEN Temperature! = VAL(NewTemp$)
 LOCATE 17, 55, 0
 PRINT Pressure!
 LOCATE 17, 10, 0
 INPUT "Please enter the atmospheric pressure (mmHg): ",
NewPres$
 IF LEN(NewPres$) > 0 THEN Pressure! = VAL(NewPres$)
 LOCATE 20, 40, 0
 PRINT Wavelength!
 LOCATE 19, 10, 0
 PRINT "Please enter the center wavelength of light"
 LOCATE 20, 10, 0
 INPUT "for the chosen star (microns): ", NewWave$
 IF LEN(NewWave$) > 0 THEN Wavelength! = VAL(NewWave$)
'
END SUB

'****************************************
'**     Name:       ChooseStar         **
'**     Type:       Subroutine         **
'**     Module:     MIRACTRL.BAS       **
'**     Language:   Quickbasic 4.50    **
'****************************************
'
'Allows choice of stars from list.  Sets active star for slew
and track
'of telescope.
'
'Example of use: StarDataEntry Star$(), RA$(), DEC$(),
Epoch$(), NewFile%, w1, w1Text$(), w1Title$, w1Prompt$,
CurrentStar%
'Parameters:   (none)
'Variables:     Star$()     A 5-element matrix of desired
stars for
'                           viewing
'               RA$()          A 5-element matrix of right
ascensions
'                           for the stars listed in Star$()
```

```
'               DEC$()      A 5-element matrix of declinations
'                             for the stars listed in Star$()
'                Epoch$()     A 5-element matrix of epochs for
the
'                           RA's and DEC's of the stars listed
in Star$()
'                g%, h%, i%  Looping indices
'                    j%              A flag activated when the
Backspace key is hit
'                 l%           A flag activated when the Shift
Tab, Up or Down
'                             Arrows are hit
'               row%, col%  The current row and column markers
for the
'                             cursor
'                Stop$        A string representing whether or
not the
'                                character entry loop is to be
exited
'                  kee%          A unique code returned by the
KeyCode% or
'                            KeyCodeNoWait% Functions for each
key hit
'                 Character$  A string read from the keyboard
buffer
'               Current$    A string representing the entry of
the current
'                            field.  This variable will become
one of the
'                                     following:  Star$(), RA$(),
DEC$(), EPOCH$()
'
'Module Level
'   DECLARATIONS:  DECLARE SUB StarDataEntry (Star$(), RA$(),
DEC$(),
'                                         Epoch$(), NewFile%, w1,
w1Text$(), w1Title$, w1Prompt$, CurrentStar%)
'
SUB ChooseStar (Star$(), RA$(), DEC$(), EPOCH$(), NewFile%, w1
AS WindowsType, w1Text$(), w1Title$, w1Prompt$, currentStar%)
'
'Initialize variables
 current$ = Star$(1)
 row% = 17
 col% = 9
 FirstRow% = 17
 LastRow% = 21
 LastCol% = 62
 FileLine% = 1
 TopFileLineShown% = 1
 currentStar% = 1
```

```
 ChangeFileLine% = FALSE
 HomeOrEnd% = FALSE
 Stops% = FALSE
 QuitEntry% = FALSE
'
'Erase Box
 COLOR WHITE, BLUE, BLUE
 FOR r% = 14 TO 21
    LOCATE r%, 6, 0
    PRINT SPACE$(68);
 NEXT r%
'
'Draw bottom edge of the status box
 LOCATE 22, 5, 0
 PRINT CHR$(200); STRING$(68, 205); CHR$(188);
'
'Place lettering
 COLOR WHITE, BLUE, BLUE
 LOCATE 14, 22, 0
 PRINT "Choose a star to view and press F10."
 LOCATE 15, 33, 0
 PRINT "RA            DEC";
 LOCATE 16, 10, 0
 PRINT "Star Name              hr:min:sec";
 LOCATE 16, 48, 0
 PRINT CHR$(248); ":':''";
 LOCATE 16, 63, 0
 PRINT "Epoch";
'
'Complete data fields by reading in one key at a time, ESC
exits
 DO
        IF TopFileLineShown% < 1 THEN TopFileLineShown% = 1
        IF TopFileLineShown% > 96 THEN TopFileLineShown% = 96
        IF FileLine% < 1 THEN FileLine% = 1
        IF FileLine% > 100 THEN FileLine% = 100
        IF row% < FirstRow% THEN row% = FirstRow%
        IF row% > LastRow% THEN row% = LastRow%
        'Update the row numbers and fields
        FOR g% = 1 TO 5
                COLOR WHITE, BLUE, BLUE
                LOCATE 16 + g%, 6, 0
                PRINT TopFileLineShown% + g% - 1
                COLOR WHITE, BLACK, BLUE
                LOCATE 16 + g%, 10, 0
                PRINT  Star$(TopFileLineShown%  +  g%  -  1)  +
SPACE$(19 - LEN(Star$(TopFileLineShown% + g% - 1)))
                LOCATE 16 + g%, 33, 0
                PRINT  RA$(TopFileLineShown%  +  g%  -  1)  +
SPACE$(10 - LEN(RA$(TopFileLineShown% + g% - 1)))
                LOCATE 16 + g%, 48, 0
```

```
                PRINT DEC$(TopFileLineShown% + g% - 1) +
SPACE$(10 - LEN(DEC$(TopFileLineShown% + g% - 1)))
                LOCATE 16 + g%, 63, 0
                PRINT EPOCH$(TopFileLineShown% + g% - 1) +
SPACE$(10 - LEN(EPOCH$(TopFileLineShown% + g% - 1)))
        NEXT g%
        COLOR BLACK, WHITE, BLUE
        LOCATE row%, 10, 0
         PRINT Star$(FileLine%) + SPACE$(19 -
LEN(Star$(FileLine%)))
        LOCATE row%, 33, 0
         PRINT RA$(FileLine%) + SPACE$(10 -
LEN(RA$(FileLine%)))
        LOCATE row%, 48, 0
         PRINT DEC$(FileLine%) + SPACE$(10 -
LEN(DEC$(FileLine%)))
        LOCATE row%, 63, 0
         PRINT EPOCH$(FileLine%) + SPACE$(10 -
LEN(EPOCH$(FileLine%)))
        '
        COLOR WHITE, BLACK, BLUE
        'Read the next character and operate on it
        DO
                LOCATE row%, col% + 1, 1
                kee% = KeyCode%(Character$)
                SELECT CASE kee%
                        CASE ENTER
                                IF row% = LastRow% AND col% =
LastCol% THEN
                                        row% = row% + 1
                                        IF row% = LastRow% +
1 THEN

TopFileLineShown% = TopFileLineShown% + 1
                                                FileLine% =
FileLine% + 1
                                        END IF
                                ELSE
                                        'no action
                                END IF
                                Stops% = TRUE
                        CASE ESCAPE
                                windows w1, w1Text$(),
w1Title$, w1Prompt$
                                IF w1.returnCode = 121 OR
w1.returnCode = 89 THEN Abort% = TRUE
                                IF Abort% = TRUE THEN

'***********************************
                                        '** System Shutdown
Procedures **
```

```
                                                '**         go here!!
        **
'*********************************
                                                WindowsPop
                                                CLS
                                                SYSTEM
                                        ELSE
                                                WindowsPop
                                        END IF
                                CASE UPARROW
                                        IF row% = FirstRow% THEN
                                                TopFileLineShown%  =
TopFileLineShown% - 1
                                        END IF
                                        row% = row% - 1
                                        ChangeFileLine% = MINUS
                                        Stops% = TRUE
                                CASE DOWNARROW
                                        IF row% = LastRow% THEN
                                                TopFileLineShown%  =
TopFileLineShown% + 1
                                        END IF
                                        row% = row% + 1
                                        ChangeFileLine% = PLUS
                                        Stops% = TRUE
                                CASE PGDOWN
                                        IF TopFileLineShown% > 95 THEN
                                               ChangeFileLine% = PLUS
* (96 - TopFileLineShown%)
                                               TopFileLineShown% = 96
                                        ELSE
                                                TopFileLineShown%  =
TopFileLineShown% + 5
                                               ChangeFileLine% = PLUS
* 5
                                        END IF
                                        Stops% = TRUE
                                CASE PGUP
                                        IF TopFileLineShown% < 6 THEN
                                                ChangeFileLine%    =
MINUS * (TopFileLineShown% - 1)
                                                TopFileLineShown% = 1
                                        ELSE
                                                TopFileLineShown%  =
TopFileLineShown% - 5
                                                ChangeFileLine%    =
MINUS * 5
                                        END IF
                                        Stops% = TRUE
                                CASE HOME
```

```
                                TopFileLineShown% = 1
                                HomeOrEnd% = 1
                                Stops% = TRUE
                        CASE ENDFILE
                                TopFileLineShown% = 96
                                HomeOrEnd% = 100
                                Stops% = TRUE
                        CASE 17408 'F10
                                currentStar% = FileLine%
                                QuitEntry% = TRUE
                                Stops% = TRUE
                        CASE ELSE

'*****************************************
                                '**
         **
                                '**             Error Handling
Needed      **
                                '**
         **

'*****************************************
                END SELECT
        LOOP UNTIL Stops% = TRUE
        Stops% = FALSE
        IF FileLine% < 1 THEN FileLine% = 1
        IF FileLine% > 100 THEN FileLine% = 100
        IF ChangeFileLine% <> FALSE THEN
                FileLine% = FileLine% + ChangeFileLine%
                IF FileLine% < 1 THEN FileLine% = 1
                IF FileLine% > 100 THEN FileLine% = 100
                ChangeFileLine% = FALSE
                current$ = Star$(FileLine%)
        END IF
        '
        IF HomeOrEnd% <> FALSE THEN
                IF HomeOrEnd% = 1 THEN
                        row% = FirstRow%
                ELSEIF HomeOrEnd% = 100 THEN
                        row% = LastRow%
                END IF
                FileLine% = HomeOrEnd%
                HomeOrEnd% = FALSE
                COLOR WHITE, BLUE, BLUE
                LOCATE LastRow%, 9, 0
                PRINT SPACE$(1)
                current$ = Star$(FileLine%)
        END IF
        '
 LOOP UNTIL QuitEntry% = TRUE
'
```

```
END SUB

'*************************************
'**     Name:        DayLightSaving    **
'**     Type:        Subroutine        **
'**     Module:      MIRACTRL.BAS      **
'**     Language:    Quickbasic 4.50   **
'*************************************
'
'Determine if Daylight Savings Time applies to local time
'
'Example of use:  DayLightSaving DayLightSavings%, hour%,
day%, month%, year%
'Parameters:      (none)
'Variables:      DayLightSavings%  A varaible representing
whether or not
'                                  daylight savings time is
in effect
'              dayOfWeek$  A string representing the day of
the week
'                  SundayPassed%  A varialbe representing
whether or not
'                                  Sunday has passed in the
current week
'                hour%      The current hour of time
'                day%       The current day
'                month%     The current month
'                year%      The current year
'
'Module Level
'
'           DECLARATIONS:     DECLARE  SUB  DayLightSaving
(DayLightSavings%, hour%, day%, month%, year%)
'
SUB DayLightSaving (DayLightSavings%, hour%, day%, month%,
year%)
        '**************************************************
        '**  Determine the day of the week (=dayOfWeek$)   **
        '**  and if Sunday has passed (=SundayPassed$)     **
        '**************************************************
        IF month% > 10 OR month% < 4 THEN DayLightSavings% =
FALSE
        IF month% > 4 AND month% < 10 THEN DayLightSavings% =
TRUE
        IF month% = 4 THEN
                IF day% > 23 THEN
                        IF dayOfWeek$ = "SUNDAY" THEN
                                IF    hour%    >    1    THEN
DayLightSavings% = TRUE
                        ELSEIF  SundayPassed%  =  TRUE  THEN
DayLightSavings% = TRUE
```

```
                END IF
            END IF
        END IF
        IF month% = 10 THEN
            IF day% > 24 THEN
                IF dayOfWeek$ = "SUNDAY" THEN
                    IF    hour%    >    1    THEN
DayLightSavings% = TRUE
                ELSEIF  SundayPassed%  =  TRUE  THEN
DayLightSavings% = TRUE
                END IF
            END IF
        END IF
END SUB

'****************************************
'**     Name:       EquationEquinoxes **
'**     Type:       Subroutine        **
'**     Module:     MIRACTRL.BAS      **
'**     Language:   Quickbasic 4.50   **
'****************************************
'
'Allows input of time correction argument from the Equation of
Equinoxes
'(See page B6-B15, Astronomical Almanac)(Correction needed due
to nutation
'of Earth)
'
'Example of use:  EquationEquinoxes EQofEQ(), EQofEQUIN!
'Parameters:      (none)
'Variables:      EQofEQ()     An array of the the equation of
equinox
'                              results for the next two days
'                      EQofEQUIN!    The equation of equinox
correction for
'                              the current time
'                  NextDay$     A string representing either
the current
'                              or next day
'                  NextDate$    A string representing either
the current
'                             or the next total date (month,
day, year)
'
'Module Level
'    DECLARATIONS:  DECLARE SUB EquationEquinoxes (EQofEQ(),
EQofEQUIN!)
'
SUB EquationEquinoxes (EQofEQ(), EQofEQUIN!)
        'Input the equation of equinox corrections for the
current day
```

```
        'and the next day
        COLOR WHITE, BLUE, BLUE
        FOR i% = 1 TO 2
                NextDay$ = LTRIM$(STR$(VAL(MID$(DATE$, 4, 2))
+ i% - 1))
                NextDate$  =  LEFT$(DATE$,  3)  +  NextDay$  +
RIGHT$(DATE$, 5)
                LOCATE 16 + i%, 57, 0
                PRINT EQofEQ(i%)
                LOCATE 16 + i%, 10, 0
                PRINT "Enter the Equation of Equinoxes for ";
NextDate$; ";";
                LOCATE 17 + i%, 10, 0
                PRINT "(in seconds)"
                LOCATE 16 + i%, 58, 1
                INPUT "", EQofEQ(i%)
        NEXT i%
        '
        'Interpolate  to  find  the  equation  of  equinox
correction for the
        'current time
        EQofEQUIN!  =  (TIMER  *  (EQofEQ(2)  -  EQofEQ(1))  /
86400!) + EQofEQ(1)
        '
        'Erase the above cues
        FOR i% = 1 TO 4
                LOCATE 16 + i%, 21, 0
                PRINT SPACE$(45)
        NEXT i%
END SUB

'****************************************
'**     Name:       InitialDisplay    **
'**     Type:       Subroutine        **
'**     Module:     MIRACTRL.BAS      **
'**     Language:   Quickbasic 4.50   **
'****************************************
'
'Creates the initial display screen graphics
'
'Example of use:  InitialDisplay
'Parameters:      (none)
'Variables:       i%, r%     Looping indices
'
'Module Level
'    DECLARATIONS:  DECLARE SUB InitialDisplay
'
 SUB InitialDisplay
 CLS
'Color the background
 FOR i% = 1 TO 25
```

```
     COLOR BLUE, BLACK, BLUE
     PRINT STRING$(80, 177)
 NEXT i%
 LOCATE 24, 1, 0
 PRINT STRING$(80, 177);
 LOCATE 25, 1, 0
 PRINT STRING$(80, 177);
'
'Place the lettering
 LOCATE 4, 18, 0
 COLOR WHITE, BLUE, BLUE
 PRINT "Monterey Institute for Research in Astronomy"
 LOCATE 6, 28, 0
 PRINT "Oliver Observing Station"
 LOCATE 10, 30, 0
 PRINT "Automated Telescope"
 LOCATE 11, 32, 0
 PRINT "Control Package"
'
'Draw top edge of status box
 LOCATE 13, 5, 0
 COLOR WHITE, BLUE, BLUE
 PRINT CHR$(201); STRING$(68, 205); CHR$(187);
'
'Draw the body of the status box
 FOR r% = 14 TO 21
    LOCATE r%, 5, 0
    PRINT CHR$(186);
    PRINT SPACE$(68);
    PRINT CHR$(186);
 NEXT r%
'
'Draw bottom edge of the status box
 LOCATE 22, 5, 0
 PRINT CHR$(200); STRING$(68, 205); CHR$(188);
'
'Display "Initializing" status in box
 COLOR WHITE + BLINK, BLUE, BLUE
 LOCATE 17, 25, 0
 PRINT "Initializing"
 LOCATE 17, 37, 0
 COLOR WHITE, BLUE, BLUE
 PRINT " - Please stand by"
END SUB

'****************************************
'**      Name:        KeyCode%         **
'**      Type:        Function         **
'**      Module:      MIRACTRL.BAS     **
'**      Language:    Quickbasic 4.50  **
'****************************************
```

```
'
'Returns a unique integer for any key pressed
'
'Example of Use: kee% = KeyCode% Character$
'Parameters:    (none)
'Variables:       Character$      A string representing the
current keyboard
'                                buffer value
'
'Module Level
'    DECLARATIONS:  DECLARE FUNCTION KeyCode% (Character$)
'
FUNCTION KeyCode% (Character$) STATIC
        DO
                Character$ = INKEY$
        LOOP UNTIL Character$ <> ""
    KeyCode% = CVI(Character$ + CHR$(0))
END FUNCTION

'***************************************
'**     Name:        KeyCodeNoWait%    **
'**     Type:        Function          **
'**     Module:      MIRACTRL.BAS      **
'**     Language:    Quickbasic 4.50   **
'***************************************
'
'Returns a unique integer for any key pressed, but does not
wait
'for a key to be pressed
'
'Example of Use: kee% = KeyCodeNoWait% Character$
'Parameters:    (none)
'Variables:     (none)
'Module Level
'    DECLARATIONS:  DECLARE FUNCTION KeyCode% (Character$)
'
FUNCTION KeyCodeNoWait% (Character$) STATIC
    Character$ = INKEY$
    IF Character$ <> "" THEN
        KeyCodeNoWait% = CVI(Character$ + CHR$(0))
    ELSE
        KeyCodeNoWait% = 0
    END IF
END FUNCTION

'***************************************
'**     Name:        Nutation          **
'**     Type:        Subroutine        **
'**     Module:      MIRACTRL.BAS      **
'**     Language:    Quickbasic 4.50   **
'***************************************
```

```
'
'Calculates the nutation corrections to celestial coordinates
for a given
'day and time.
'
'Example of use: Nutation RAofStar#, DECofStar#, JulianDate#,
N#(), S1#(), S2#()
'
'Note: Astronomical Almanac notation is used wherever possible
'
SUB Nutation (RAofStar#, DECofStar#, JulianDate#, N#(), S1#(),
S2#())
'
'Unmodified celestial coordinates converted to rectangular
form
 S1#(1) = COS(DECofStar#) * COS(RAofStar#)
 S1#(2) = COS(DECofStar#) * SIN(RAofStar#)
 S1#(3) = SIN(DECofStar#)
'
'Nutation Parameters
 d# = JulianDate# - 2447871.5#
 epsilon# = 23.440332# * (2# * pi) / 360#
'
 DeltaPsi# = -.0048# * SIN((279.9# - .053# * d#) * (2# * pi)
/ 360#) - .0004# * SIN((197.9# + 1.971# * d#) * (2# * pi) /
360#)
 DeltaPsi# = DeltaPsi# * (2# * pi) / 360#
'
 DeltaEpsilon# = .0026# * COS((279.9# - .053# * d#) * (2# *
pi) / 360#) + .0002# * COS((197.9# + 1.971# * d#) * (2# * pi)
/ 360#)
 DeltaEpsilon# = DeltaEpsilon# * (2# * pi) / 360#
'
'Nutation Rotation Matrix
 N#(1, 1) = 1
 N#(1, 2) = -DeltaPsi * COS(epsilon#)
 N#(1, 3) = -DeltaPsi * SIN(epsilon#)
 N#(2, 1) = DeltaPsi * COS(epsilon#)
 N#(2, 2) = 1
 N#(2, 3) = -DeltaEpsilon#
 N#(3, 1) = DeltaPsi * SIN(epsilon#)
 N#(3, 2) = DeltaEpsilon#
 N#(3, 3) = 1
'
'Celestial Coordinates modified for nutation in rectangular
form
 S2#(1) = N#(1, 1) * S1#(1) + N#(1, 2) * S1#(2) + N#(1, 3) *
S1#(3)
 S2#(2) = N#(2, 1) * S1#(1) + N#(2, 2) * S1#(2) + N#(2, 3) *
S1#(3)
```

```
 S2#(3) = N#(3, 1) * S1#(1) + N#(3, 2) * S1#(2) + N#(3, 3) *
S1#(3)
'
'Decode S1#() elements from rectangular form to RA and DEC
 IF S2#(1) >= 0# AND S2#(2) < 0# THEN
        alpha# = ATN(-S2#(1) / S2#(2)) + (3# * pi) / 2#
 ELSEIF S2#(1) < 0# AND S2#(2) < 0# THEN
        alpha# = ATN(S2#(2) / S2#(1)) + pi
 ELSEIF S2#(1) < 0# AND S2#(2) >= 0# THEN
        alpha# = ATN(-S2#(1) / S2#(2)) + pi / 2#
 ELSE alpha# = ATN(S2#(2) / S2#(1))
 END IF
'
 delta# = ATN(S2#(3) * COS(alpha#) / S2#(1))
'
 RAofStar# = alpha#
 DECofStar# = delta#
'
END SUB

'****************************************
'**     Name:        Precession        **
'**     Type:        Subroutine        **
'**     Module:      MIRACTRL.BAS      **
'**     Language:    Quickbasic 4.50   **
'****************************************
'
'Calculates the precession corrections to celestial
coordinates for a given
'day and time.
'
'Example of use: Precession RA$(), DEC$(), EPOCH$(),
currentStar%, JulianDate#, EpochError%, RAofStar#, DECofStar#,
RADecError%
'
'Note: Astronomical Almanac notation is used wherever possible
'
SUB Precession (RA$(), DEC$(), EPOCH$(), currentStar%,
JulianDate#, EpochError%, RAofStar#, DECofStar#, RADecError%)
'
'Decode RA$() and DEC$() into radians
 RAFirstColon% = INSTR(RA$(currentStar%), ":")
 RASecondColon% = INSTR((RAFirstColon% + 1),
RA$(currentStar%), ":")
 IF RAFirstColon% = 0 OR RASecondColon% = 0 THEN
        RADecError% = TRUE
 ELSE
         RAofStar# = VAL(LEFT$(RA$(currentStar%),
(RAFirstColon% - 1))) * 3600 + VAL(MID$(RA$(currentStar%),
(RAFirstColon% + 1), 2)) * 60 + VAL(RIGHT$(RA$(currentStar%),
(LEN(RA$(currentStar%)) - RASecondColon%)))
```

```
        RAofStar# = RAofStar# * 2# * pi / 86164.09053#
'Seconds of time converted to radians of sidereal movement
        DO WHILE RAofStar# >= (2# * pi)
                RAofStar# = RAofStar# - (2# * pi)
        LOOP
        DO WHILE RAofStar# < 0
                RAofStar# = RAofStar# + (2# * pi)
        LOOP
 END IF
 DECFirstColon% = INSTR(DEC$(currentStar%), ":")
 DECSecondColon% = INSTR((DECFirstColon% + 1),
DEC$(currentStar%), ":")
 IF DECFirstColon% = 0 OR DECSecondColon% = 0 THEN
        RADecError% = TRUE
 ELSE
         DECofStar# = VAL(LEFT$(DEC$(currentStar%),
(DECFirstColon% - 1))) + VAL(MID$(DEC$(currentStar%),
(DECFirstColon% + 1), 2)) / 60 +
VAL(RIGHT$(DEC$(currentStar%), (LEN(RA$(currentStar%)) -
DECFirstColon%))) / 3600
        DECofStar# = DECofStar# * (2# * pi) / 360#'Degrees
converted to radians
        DO WHILE DECofStar# > (pi / 2#)
                DECofStar# = pi - DECofStar#
        LOOP
        DO WHILE DECofStar# < -(pi / 2#)
                DECofStar# = -(pi + DECofStar#)
        LOOP
 END IF
'Account for precession between the epoch chosen by the user
and J2000.0
'
'Determine the Julian date at last midnight
 year! = VAL(EPOCH$(currentStar%))
 IF year! = 0 THEN
        EpochError% = TRUE
 ELSE
        JDofEpoch! = (4713! + year!) * 365.25
        '
        'Using the approximate formulae for precession in the
Astronomical Almanac:
        'Account for precession FROM the star's known epoch TO
J2000.0
        T1# = (CDBL(JDofEpoch!) - 2451545#) / 36525#
        M1# = 1.2812323# * T1# + .0003879# * T1# ^ 2 +
.0000101# * T1# ^ 3'  degrees
        N1# = .5567530000000001# * T1# - .0001185# * T1# ^ 2
- .0000116# * T1# ^ 3'   degrees
        '
        alpha# = RAofStar# * 360# / (2# * pi)'   RA in degrees
```

```
        delta# = DECofStar# * 360# / (2# * pi)'        Dec in
degrees
        alphaInRad# = RAofStar#
        deltaInRad# = DECofStar#
        '
        'M suffix refers to mean epoch
        alphaM# = alpha# - .5# * (M1# - N1# * SIN(alphaInRad#)
* TAN(deltaInRad#))
        alphaMinRad# = alphaM# * (2# * pi) / 360#
        deltaM# = delta# - .5# * N1# * COS(alphaMinRad#)
        deltaMinRad# = deltaM# * (2# * pi) / 360#
        '
        '0 suffix refers to J2000.0 epoch
        alpha0# = alpha# - M1# - N1# * SIN(alphaMinRad#) *
TAN(deltaMinRad#)
        alpha0InRad# = alpha0# * (2# * pi) / 360#
        delta0# = delta# - N1# * COS(alphaMinRad#)
        delta0InRad# = delta0# * (2# * pi) / 360#
        '
        'Account for precession FROM J2000.0 TO the current
date
        T2# = (CDBL(JulianDate#) - 2451545#) / 36525#
        M2# = 1.2812323# * T2# + .0003879# * T2# ^ 2 +
.0000101# * T2# ^ 3'  degrees
        N2# = .5567530000000001# * T2# - .0001185# * T2# ^ 2
- .0000116# * T2# ^ 3'   degrees
        '
        'M suffix refers to mean epoch
        alphaM#   =   alpha0#   -   .5#   *   (M2#   -   N2#   *
SIN(alpha0InRad#) * TAN(delta0InRad#))
        alphaMinRad# = alphaM# * (2# * pi) / 360#
        deltaM# = delta0# - .5# * N2# * COS(alphaMinRad#)
        deltaMinRad# = deltaM# * (2# * pi) / 360#
        '
        'The new alpha and delta are the CORRECTED right
ascension and
        'declination IN DEGREES
        alpha# = alpha0# - M2# - N2# * SIN(alphaMinRad#) *
TAN(deltaMinRad#)
        delta# = delta0# - N2# * COS(alphaMinRad#)
        RAofStar# = alpha# * (2# * pi) / 360#
        DO WHILE RAofStar# >= (2# * pi)
                RAofStar# = RAofStar# - (2# * pi)
        LOOP
        DO WHILE RAofStar# < 0
                RAofStar# = RAofStar# + (2# * pi)
        LOOP
        '
        DECofStar# = delta# * (2# * pi) / 360#
        DO WHILE DECofStar# > (pi / 2#)
                DECofStar# = pi - DECofStar#
```

```
        LOOP
        DO WHILE DECofStar# < -(pi / 2#)
                DECofStar# = -(pi + DECofStar#)
        LOOP
        '
 END IF
END SUB

'*****************************************
'**     Name:       Refraction         **
'**     Type:       Subroutine         **
'**     Module:     MIRACTRL.BAS       **
'**     Language:   Quickbasic 4.50    **
'*****************************************
'
'Calculates the refraction corrections to celestial coordinates for a given
'day and time.  Uses data from the Atmospheric Subroutine.
'
'Example of use:  Refraction Pressure!, Temperature!, Wavelength!, GMST0hrSEC#, HourAngle#, GHAaries#, RAofStar#, DECofStar#, RARefractCor#, DecRefractCor#, RAofMIRA#
'
SUB Refraction (Pressure!, Temperature!, Wavelength!, GMST0hrSEC#, HourAngle#, GHAaries#, RAofStar#, DECofStar#, RARefractCor#, DecRefractCor#, RAofMIRA#)
'
'Initialize variables
 MIRAlatitude# = .63908139#        'radians, (36 deg 37 min North)
 MIRAlongitude# = 2.12639281#      'radians, (121 deg 50 min West)
 sec% = VAL(RIGHT$(TIME$, 2))
 min% = VAL(MID$(TIME$, 4, 2))
 hour% = VAL(LEFT$(TIME$, 2))
'
'Calculate Greenwhich Mean Sidereal Time
 'Determine the day fraction that has passed since midnight
 'in SECONDS
 sec& = sec%
 hours& = hour%
 mins& = min%
 dayFraction# = CDBL((hours& * 3600) + (mins& * 60) + sec&)
 '
 'Determine the Greenwhich Mean Sidereal Time at the current
 'time in SECONDS
 GMSTinSEC# = GMST0hrSEC# + dayFraction#
'
'Calculate Greenwhich Hour Angle of Aries
 GHAaries# = GMSTinSEC# * 2# * pi / 86164.09053099999#'Seconds of solar time converted to radians
```

```
 DO WHILE GHAaries# > (2# * pi)
        GHAaries# = GHAaries# - (2# * pi)
 LOOP
 DO WHILE GHAaries# < 0#
        GHAaries# = GHAaries# + (2# * pi)
 LOOP
'
'Calculate the current star's hour angle for this moment
 RAofMIRA# = GHAaries# - MIRAlongitude#
'
 HourAngle# = RAofMIRA# - RAofStar#   'in radians
 DO WHILE HourAngle# > (2# * pi)
        HourAngle# = HourAngle# - (2# * pi)
 LOOP
 DO WHILE HourAngle# < 0#
        HourAngle# = HourAngle# + (2# * pi)
 LOOP
'
'Calculate the refraction constant in seconds
 k# = 21.3 * Pressure! * (1 + (.0057 / (Wavelength! ^ 2))) /
(273 + Temperature!)
'Note:  Temperature in degrees C, Pressure in mmHg, Wavelength
in microns
'       K# in seconds of arc
'
'Calculate the refraction corrections in seconds
 RARefractCor# = k# * SIN(HourAngle#) / ((COS(DECofStar#) ^ 2)
* (TAN(DECofStar#) * TAN(MIRAlatitude#) + COS(HourAngle#)))
 DecRefractCor# = k# * (TAN(DECofStar#) * COS(HourAngle#) -
TAN(MIRAlatitude#)) / (TAN(DECofStar#) * TAN(MIRAlatitude#) +
COS(HourAngle#))
 RARefractCor# = RARefractCor# * (2# * pi) / 3600#'  Convert
seconds of arc to radians
 DecRefractCor# = DecRefractCor# * (2# * pi) / 3600#'  Convert
seconds of arc to radians

'
END SUB

'****************************************
'**     Name:       SetCommands       **
'**     Type:       Subroutine        **
'**     Module:     MIRACTRL.BAS      **
'**     Language:   Quickbasic 4.50   **
'****************************************
'
'Controls motor commands to set the Telescope following a slew
to a new
'target star.  Uses the stepping motors in high speed mode.
'
```

```
'Example of use:  SetCommands Simulation%, TeleRA#, TeleDec#,
RAofStar#, DECofStar#
'
SUB SetCommands (Simulation%, TeleRA#, TeleDEC#, RAofStar#,
DECofStar#)
'
'Set as appropriate in Right Ascension
 DO
        RAdiff# = TeleRA# - RAofStar#
        'Update motor display box
        COLOR RED, BLUE, BLUE
        LOCATE 3, 20, 0
        PRINT CHR$(222)
        'POKE (address of RA Set controller), X
        '***********************
        IF Simulation% = TRUE THEN
                IF RAdiff# > 0 THEN TeleRA# = TeleRA# - .00001
                IF RAdiff# < 0 THEN TeleRA# = TeleRA# + .00001
        ELSE
                'Read new TeleRA# from encoders
        END IF
        '
        'Check limit switch
        'PEEK (address of limit switch)
        '*******************************
        '** Change RAdiff# < .0003 to  **
        '** acceptable limit           **
 LOOP UNTIL RAdiff# < .0003
        '*******************************
'
'Set as appropriate in Declination
 DO
        DECdiff# = TeleDEC# - DECofStar#
        'Update motor display box
        COLOR RED, BLUE, BLUE
        LOCATE 3, 70, 0
        PRINT CHR$(222)
        'POKE (address of DEC Set controller), X
        '***********************
        IF Simulation% = TRUE THEN
                IF DECdiff# > 0 THEN TeleDEC# = TeleDEC# -
.00001
                IF DECdiff# < 0 THEN TeleDEC# = TeleDEC# +
.00001
        ELSE
                'Read new TeleDEC# from encoders
        END IF
        '
        'Check limit switch
        'PEEK (address of limit switch)
        '*******************************
```

```
        '** Change DECdiff# < .0003 to  **
        '** acceptable limit            **
 LOOP UNTIL DECdiff# < .0003
        '********************************
'
END SUB

'****************************************
'**     Name:       ShowTime            **
'**     Type:       Subroutine          **
'**     Module:     MIRACTRL.BAS        **
'**     Language:   Quickbasic 4.50     **
'****************************************
'
'Allows entry of star position data as an initial user input
'
'Example of use: ShowTime min%, hour%, day%, month%, year%,
DayLightSavings%, SiderealTime$
'Parameters:
'Variables:      SiderealTime$   A string representing the
local
'                                apparent sidereal time
'                DayLightSavings%  A variable showing the
state of daylight
'                                 savings time - on or off
'               min%          The current minute of time
'               hour%         The current hour of time
'               day%          The current day
'               month%        The current month
'               year%         The current year
'
'Module Level
'   DECLARATIONS: DECLARE SUB ShowTime (min%, hour%, day%,
month%, year%, DayLightSavings%, SiderealTime$)
'
SUB   ShowTime   (min%,   hour%,   day%,   month%,   year%,
DayLightSavings%, SiderealTime$)
        'Display  local  time  in  accordance  with  daylight
savings time.
        IF DayLightSavings% = FALSE THEN
                LocalTime$ = STR$(VAL(LEFT$(TIME$, 2)) - 5) +
RIGHT$(TIME$, 6)
        ELSE
                LocalTime$ = STR$(VAL(LEFT$(TIME$, 2)) - 4) +
RIGHT$(TIME$, 6)
        END IF
        '
        IF VAL(LEFT$(LocalTime$, 2)) < 1 THEN
                LocalTime$ = STR$(VAL(LEFT$(LocalTime$, 2)) +
24) + RIGHT$(LocalTime$, 6)
        END IF
```

```
        '
        FirstColonLoc% = INSTR(SiderealTime$, ":")
        LastColonLoc%    =    INSTR((FirstColonLoc%    +    1),
SiderealTime$, ":")
        LengthSec% = 2
        BeginMin% = FirstColonLoc% + 1
        '
        SiderealSec$ = LTRIM$(STR$(VAL(RIGHT$(SiderealTime$,
LengthSec%)) + 1))
        SiderealMin$    =    MID$(SiderealTime$,    BeginMin%,
LastColonLoc% - FirstColonLoc% - 1)
        SiderealHr$ = LEFT$(SiderealTime$, (FirstColonLoc% -
1))
        '
        IF LEN(SiderealSec$) = 1 THEN SiderealSec$ = "0" +
SiderealSec$
        IF VAL(SiderealSec$) > 59 THEN
                SiderealMin$ = STR$(VAL(SiderealMin$) + 1)
                SiderealSec$ = "00"
        END IF
        IF VAL(SiderealMin$) > 59 THEN
                SiderealHr$ = STR$(VAL(SiderealHr$) + 1)
                SiderealMin$ = "00"
        END IF
        IF VAL(SiderealHr$) > 23 THEN SiderealHr$ = "00"
        SiderealTime$    =    LTRIM$(SiderealHr$)    +    ":"    +
LTRIM$(SiderealMin$) + ":" + LTRIM$(SiderealSec$)
        COLOR WHITE, BLUE, BLUE
        LOCATE 15, 58, 0
        PRINT LTRIM$(LocalTime$)
        LOCATE 18, 58, 0
        PRINT LTRIM$(SiderealTime$); SPACE$(1)
        LOCATE 21, 58, 0
        PRINT TIME$
END SUB

'****************************************
'**     Name:       SiderealTimeCalc   **
'**     Type:       Subroutine         **
'**     Module:     MIRACTRL.BAS       **
'**     Language:   Quickbasic 4.50    **
'****************************************
'
'Calculate Local Apparent Sidereal Time
'
'Example of use: SiderealTimeCalc SiderealTime$, min%, hour%,
day%, month%, year%, EQofEQUIN!, GMST0hrSEC#, JulianDate#,
dayFraction#, DaysPassed%
'Parameters:     (none)
'Variables:       SiderealTime$    A string representing the
local
```

```
'                                      apparent sidereal time
'                   EQofEQ!()            The Equation of Equinox
corrections to
'                                     sidereal time for successive
dates (due
'                                   to corrections for the Earth's
nutation
'                min%             The current minute of time
'                hour%            The current hour of time
'                day%             The current day
'                month%           The current month
'                year%            The current year
'                  yearDiff!         The number of years passed
since a known
'                                    leap year (1992) divided by 4
'                  yearDiffStrg$ The decimal portion of the
above in string
'                                    form.  yearDiff! then becomes
the value of
'                                    this string
'               LeapYear$        A string indicating whether or
not the
'                                    current year is a leap year
'                noonPassed$      A string representing whether
or not noon
'                                    has passed on the current day
'                 daysPassed%      The number of days passed in
the current
'                                    year
'                  JulianDate#      The current Julian date in
seconds
'                    Tu#                  A variable used in the
calculation of
'                                    Greenwhich Mean Sidereal Time
at 0 hour,
'                                  following the convention in the
Astronomical
'                                    Almanac
'                GMST0hrSEC!      Greenwhich Mean Sidereal Time
at 0 hour on
'                                    the current day
'                 dayFraction!   The decimal fraction of the
current day that
'                                   has passed by the current time
'                GMSTinSEC!       Greenwhich Mean Sidereal Time
at the currrent
'                                    time in seconds
'                OKtoQUIT$        A string that becomes of some
length when a
'                                    loop is to be exited, then is
reset
```

```
'
'Module Level
'           DECLARATIONS:     DECLARE   SUB   SiderealTimeCalc
(SiderealTime$, min%, hour%, day%, month%, year%, EQofEQUIN!,
GMST0hrSEC#,    GMST0hrSEC#,    JulianDate#,    dayFraction#,
DaysPassed%)
'
 SUB SiderealTimeCalc (SiderealTime$, min%, hour%, day%,
month%,   year%,   EQofEQUIN!,   GMST0hrSEC#,   JulianDate#,
dayFraction#, DaysPassed%)
        'Determine if the current year is a leap year
        yearDiff! = CSNG(year% - 1992) / 4!
         yearDiffStrg$    =    RIGHT$(STR$(yearDiff!),
(LEN(yearDiff!) - INSTR(STR$(yearDiff%), ".")))
        yearDiff! = VAL(yearDiffStrg$)
        IF yearDiff! > 0 THEN
                LeapYear$ = "FALSE"
        ELSE
                LeapYear$ = "TRUE"
        END IF
        '
        'Determine if noon has passed on the current day
        IF hour% < 12 THEN
                noonPassed$ = "FALSE"
        ELSE
                noonPassed$ = "TRUE"
        END IF
        'Determine days that have passed in the current year
        SELECT CASE month%
                CASE 1
                        DaysPassed% = day% - 1
                CASE 2
                        DaysPassed% = day% + 30
                CASE 3
                        IF LeapYear$ = "TRUE" THEN
                                DaysPassed% = day% + 59
                        ELSE
                                DaysPassed% = day% + 58
                        END IF
                CASE 4
                        IF LeapYear$ = "TRUE" THEN
                                DaysPassed% = day% + 90
                        ELSE
                                DaysPassed% = day% + 89
                        END IF
                CASE 5
                        IF LeapYear$ = "TRUE" THEN
                                DaysPassed% = day% + 120
                        ELSE
                                DaysPassed% = day% + 119
                        END IF
```

```
        CASE 6
                IF LeapYear$ = "TRUE" THEN
                        DaysPassed% = day% + 151
                ELSE
                        DaysPassed% = day% + 150
                END IF
        CASE 7
                IF LeapYear$ = "TRUE" THEN
                        DaysPassed% = day% + 181
                ELSE
                        DaysPassed% = day% + 180
                END IF
        CASE 8
                IF LeapYear$ = "TRUE" THEN
                        DaysPassed% = day% + 212
                ELSE
                        DaysPassed% = day% + 211
                END IF
        CASE 9
                IF LeapYear$ = "TRUE" THEN
                        DaysPassed% = day% + 243
                ELSE
                        DaysPassed% = day% + 242
                END IF
        CASE 10
                IF LeapYear$ = "TRUE" THEN
                        DaysPassed% = day% + 273
                ELSE
                        DaysPassed% = day% + 272
                END IF
        CASE 11
                IF LeapYear$ = "TRUE" THEN
                        DaysPassed% = day% + 304
                ELSE
                        DaysPassed% = day% + 303
                END IF
        CASE 12
                IF LeapYear$ = "TRUE" THEN
                        DaysPassed% = day% + 334
                ELSE
                        DaysPassed% = day% + 333
                END IF
        CASE ELSE
END SELECT
'
'Determine the Julian date at last midnight
'*****************************************************
'**  Change to be good for all years, not just 1992 **
JulianDate# = CDBL(2448621.5# + DaysPassed% + 1)   '**
'*****************************************************
'
```

```
        'Determine the Greenwhich Mean Sidereal Time at 0 hour
UTC
        'in SECONDS
        Tu# = (JulianDate# - 2451545#) / 36525!
        GMST0hrSEC# = 24110.54841# + (8640184.812866# * Tu#)
+ (.093104 * (Tu) ^ 2) - (6.2 * (10 ^ (-6)) * (Tu) ^ 3)
        '
        'Determine the day fraction that has passed since
midnight
        'in SECONDS
        hours& = hour%
        mins& = min%
        dayFraction# = CDBL((hours& * 3600) + (mins& * 60))
        '
        'Determine the Greenwhich Mean Sidereal Time at the
current
        'time in SECONDS
        GMSTinSEC# = GMST0hrSEC# + dayFraction#
        '
        'Calculate Greenwhich Apparent Sidereal Time in
SECONDS
        GASTinSEC# = GMSTinSEC# + CDBL(EQofEQ!)
        'Add or subtract multiples of 24 hours as necessary
        DO
                IF GASTinSEC# < 0 THEN
                GASTinSEC# = GASTinSEC# + (24! * 3600!)
                ELSEIF GASTinSEC# > 86400 THEN
                GASTinSEC# = GASTinSEC# - (24 * 3600)
                END IF
                IF GASTinSEC# >= 0 AND GASTinSEC# <= 86400
THEN OKtoQUIT% = TRUE
        LOOP UNTIL OKtoQUIT% = TRUE
        OKtoQUIT% = TRUE
        '
        GASThr& = INT(GASTinSEC# / 3600)
        GASTmin% = INT((GASTinSEC# - (GASThr& * 3600)) / 60)
        GASTsec% = INT(GASTinSEC# - (GASThr& * 3600) -
(GASTmin% * 60))
        SiderealTime$ = LTRIM$(STR$(GASThr&)) + ":" +
LTRIM$(STR$(GASTmin%)) + ":" + LTRIM$(STR$(GASTsec%))
END SUB

'****************************************
'**     Name:        SlewCommands      **
'**     Type:        Subroutine        **
'**     Module:      MIRACTRL.BAS      **
'**     Language:    Quickbasic 4.50   **
'****************************************
'
'Controls motor commands to slew the Telescope to a new
'target star.  Uses the slew motors.
```

```
'
'Example  of  use:   SlewCommands  (Simulation%,  TeleRA#,
TeleDec#, RAofStar#, DECofStar#)
'
SUB SlewCommands (Simulation%, TeleRA#, TeleDEC#, RAofStar#,
DECofStar#)
'
'Slew as appropriate in Right Ascension
 DO
        RAdiff# = TeleRA# - RAofStar#
        DO WHILE RAdiff# >= (2# * pi)
                RAdiff# = RAdiff# - (2# * pi)
        LOOP
        DO WHILE RAdiff# <= -(2# * pi)
                RAdiff# = RAdiff# + (2# * pi)
        LOOP
        'Update motor display box
        COLOR RED, BLUE, BLUE
        LOCATE 3, 10, 0
        PRINT CHR$(222)
        'Turn RA slew motor ON
        'POKE (address of RA slew controller), X
        '**********************
        IF Simulation% = TRUE THEN
                IF RAdiff# > 0 THEN TeleRA# = TeleRA# - .0001
                IF RAdiff# < 0 THEN TeleRA# = TeleRA# + .0001
        ELSE
                'Read new TeleRA# from encoders
        END IF
        '
        'Check limit switch
        'PEEK (address of limit switch)
        '********************************
        '** Change RAdiff# < .0003 to  **
        '** acceptable limit           **
 LOOP UNTIL RAdiff# < .0003
        '********************************
        'Update motor display box
        COLOR RED, BLUE, BLUE
        LOCATE 3, 10, 0
        PRINT SPACE$(1)
'
'Slew as appropriate in Declination
 DO
        DECdiff# = TeleDEC# - DECofStar#
        'Update motor display box
        COLOR RED, BLUE, BLUE
        LOCATE 3, 60, 0
        PRINT CHR$(222)
        'Turn DEC slew motor ON
        'POKE (address of DEC slew controller), X
```

```
        '************************
        IF Simulation% = TRUE THEN
                IF DECdiff# > 0 THEN TeleDEC# = TeleDEC# -
.0001
                IF DECdiff# < 0 THEN TeleDEC# = TeleDEC# +
.0001
        ELSE
                'Read new TeleDEC# from encoders
        END IF
        '
        'Check limit switch
        'PEEK (address of limit switch)
        '********************************
        '** Change DECdiff# < .0003 to  **
        '** acceptable limit            **
 LOOP UNTIL DECdiff# < .0003
        '********************************
        'Update motor display box
        COLOR RED, BLUE, BLUE
        LOCATE 3, 60, 0
        PRINT SPACE$(1)
'
END SUB

'****************************************
'**     Name:       StarDataEntry      **
'**     Type:       Subroutine         **
'**     Module:     MIRACTRL.BAS       **
'**     Language:   Quickbasic 4.50    **
'****************************************
'
'Allows entry of star position data as an initial user input
'
'Example of use: StarDataEntry Star$(), RA$(), DEC$(),
Epoch$(), NewFile%, w1, w1Text$(), w1Title$, w1Prompt$
'Parameters:    (none)
'Variables:      Star$()      A 5-element matrix of desired
stars for
'                             viewing
'                 RA$()        A 5-element matrix of right
ascensions
'                             for the stars listed in Star$()
'              DEC$()       A 5-element matrix of declinations
'                             for the stars listed in Star$()
'               Epoch$()    A 5-element matrix of epochs for
the
'                            RA's and DEC's of the stars listed
in Star$()
'               g%, h%, i%  Looping indices
'                  j%            A flag activated when the
Backspace key is hit
```

```
'                    l%            A flag activated when the Shift
Tab, Up or Down
'                                  Arrows are hit
'                 row%, col%  The current row and column markers
for the
'                                  cursor
'                   Stop$          A string representing whether or
not the
'                                     character entry loop is to be
exited
'                     kee%           A unique code returned by the
KeyCode% or
'                                 KeyCodeNoWait% Functions for each
key hit
'                    Character$  A string read from the keyboard
buffer
'                 Current$     A string representing the entry of
the current
'                                 field.  This variable will become
one of the
'                                        following:   Star$(), RA$(),
DEC$(), EPOCH$()
'
'Module Level
'    DECLARATIONS:  DECLARE SUB StarDataEntry (Star$(), RA$(),
DEC$(),
'                                                Epoch$(), NewFile%, w1,
w1Text$(), w1Title$, w1Prompt$)
'
SUB StarDataEntry (Star$(), RA$(), DEC$(), EPOCH$(), NewFile%,
w1 AS WindowsType, w1Text$(), w1Title$, w1Prompt$)
'
'Initialize variables
 IF NewFile% = TRUE THEN
        FOR h% = 1 TO 100
                Star$(h%) = ""
                RA$(h%) = ""
                DEC$(h%) = ""
                EPOCH$(h%) = ""
        NEXT h%
 END IF
 current$ = Star$(1)
 j% = 0
 L% = 0
 i% = 1
 row% = 17
 col% = 9
 InsertOn% = FALSE
 FirstRow% = 17
 LastRow% = 21
 LastCol% = 62
```

```
 FileLine% = 1
 TopFileLineShown% = 1
 ChangeCol% = FALSE
 ChangeFileLine% = FALSE
 HomeOrEnd% = FALSE
 Stops% = FALSE
 QuitEntry% = FALSE
'
'Draw bottom edge of the status box
 LOCATE 22, 5, 0
 PRINT CHR$(200); STRING$(68, 205); CHR$(188);
'
'Complete data fields by reading in one key at a time, ESC
exits
 DO
        IF TopFileLineShown% < 1 THEN TopFileLineShown% = 1
        IF TopFileLineShown% > 96 THEN TopFileLineShown% = 96
        IF FileLine% < 1 THEN FileLine% = 1
        IF FileLine% > 100 THEN FileLine% = 100
        IF row% < FirstRow% THEN row% = FirstRow%
        IF row% > LastRow% THEN row% = LastRow%
        'Update the row numbers and fields
        FOR g% = 1 TO 5
                COLOR WHITE, BLUE, BLUE
                LOCATE 16 + g%, 6, 0
                PRINT TopFileLineShown% + g% - 1
                COLOR WHITE, BLACK, BLUE
                LOCATE 16 + g%, 10, 0
                PRINT Star$(TopFileLineShown% + g% - 1) +
SPACE$(19 - LEN(Star$(TopFileLineShown% + g% - 1)))
                LOCATE 16 + g%, 33, 0
                PRINT RA$(TopFileLineShown% + g% - 1) +
SPACE$(10 - LEN(RA$(TopFileLineShown% + g% - 1)))
                LOCATE 16 + g%, 48, 0
                PRINT DEC$(TopFileLineShown% + g% - 1) +
SPACE$(10 - LEN(DEC$(TopFileLineShown% + g% - 1)))
                LOCATE 16 + g%, 63, 0
                PRINT EPOCH$(TopFileLineShown% + g% - 1) +
SPACE$(10 - LEN(EPOCH$(TopFileLineShown% + g% - 1)))
        NEXT g%
        COLOR WHITE, BLACK, BLUE
        'Read the next character and operate on it
        DO
                LOCATE row%, col% + i%, 1
                IF i% = 0 THEN
                        i% = 1
                        j% = 0
                END IF
                IF j% = 1 THEN
                        PRINT " "
                        IF LEN(current$) > 1 THEN
```

```
                                current$  =  LEFT$(current$,
(LEN(current$) - 1))
                        ELSE
                                current$ = ""
                        END IF
                        j% = 0
                END IF
                LOCATE row%, col% + i%, 1
                kee% = KeyCode%(Character$)
                SELECT CASE kee%
                        CASE ENTER
                                IF row% = LastRow% AND col% =
LastCol% THEN
                                        row% = row% + 1
                                        IF row% = LastRow% +
1 THEN

TopFileLineShown% = TopFileLineShown% + 1
                                                FileLine%  =
FileLine% + 1
                                        END IF
                                ELSE
                                        'no action
                                END IF
                                IF col% = 9 THEN
                                        i% = 20
                                ELSE
                                        i% = 11
                                END IF
                                Stops% = TRUE
                        CASE TABHIT
                                IF col% = 9 THEN
                                        i% = 20
                                ELSE
                                        i% = 11
                                END IF
                        CASE ESCAPE
                                windows   w1,   w1Text$(),
w1Title$, w1Prompt$
                                IF  w1.returnCode  =  121  OR
w1.returnCode = 89 THEN Abort% = TRUE
                                IF Abort% = TRUE THEN

'**********************************
                                        '**  System Shutdown
Procedures  **
                                        '**        go here!!
         **

'**********************************
                                        WindowsPop
```

```
                                        CLS
                                        SYSTEM
                                ELSE
                                        WindowsPop
                                END IF
                        CASE BACKSPACE
                                i% = i% - 1
                                j% = 1
                        CASE SHIFTTAB
                                IF col% = 32 THEN col% = 9
                                IF col% = 47 THEN col% = 32
                                IF col% = 62 THEN col% = 47
                                Stops% = TRUE
                        CASE UPARROW
                                IF row% = FirstRow% THEN
                                        TopFileLineShown%  =
TopFileLineShown% - 1
                                ELSE
                                        row% = row% - 1
                                END IF
                                ChangeFileLine% = MINUS
                                Stops% = TRUE
                        CASE DOWNARROW
                                IF row% = LastRow% THEN
                                        TopFileLineShown%  =
TopFileLineShown% + 1
                                ELSE
                                        row% = row% + 1
                                END IF
                                ChangeFileLine% = PLUS
                                Stops% = TRUE
                        CASE PGDOWN
                                IF TopFileLineShown% > 95 THEN
                                       ChangeFileLine% = PLUS
* (96 - TopFileLineShown%)
                                       TopFileLineShown% = 96
                                ELSE
                                        TopFileLineShown%  =
TopFileLineShown% + 5
                                       ChangeFileLine% = PLUS
* 5
                                END IF
                                Stops% = TRUE
                        CASE PGUP
                                IF TopFileLineShown% < 6 THEN
                                        ChangeFileLine%    =
MINUS * (TopFileLineShown% - 1)
                                        TopFileLineShown% = 1
                                ELSE
                                        TopFileLineShown%  =
TopFileLineShown% - 5
```

```
                                                  ChangeFileLine%     =
MINUS * 5
                                          END IF
                                          Stops% = TRUE
                                  CASE HOME
                                          TopFileLineShown% = 1
                                          HomeOrEnd% = 1
                                          Stops% = TRUE
                                  CASE ENDFILE
                                          TopFileLineShown% = 96
                                          HomeOrEnd% = 100
                                          Stops% = TRUE
                                  CASE LEFTARROW
                                          i% = i% - 1
                                  CASE RIGHTARROW
                                          IF LEN(current$) > i% THEN
                                                  i% = i% + 1
                                          END IF
                                  CASE DELETE
                                          IF   i%   >   1   AND   i%   <
LEN(current$) + 1 THEN
                                                       current$     =
LEFT$(current$, i% - 1) + RIGHT$(current$, LEN(current$) - i%)
                                          ELSEIF  i%  =  1  AND  i%  <
LEN(current$) + 1 THEN
                                                       current$     =
RIGHT$(current$, LEN(current$) - i%)
                                          ELSEIF i% = 1 THEN
                                                  current$ = ""
                                          ELSE
                                                       current$     =
LEFT$(current$, i% - 1)
                                          END IF
                                          LOCATE row%, col% + 1, 1
                                          IF col% = 9 THEN
                                                   PRINT    current$;
SPACE$(19 - LEN(current$))
                                          ELSE
                                                   PRINT    current$;
SPACE$(10 - LEN(current$))
                                          END IF
                                  CASE INSERT
                                          IF InsertOn% = TRUE THEN
                                                  InsertOn% = FALSE
                                                  LOCATE , , , 7, 7
                                          ELSE
                                                  InsertOn% = TRUE
                                                  LOCATE , , , 3, 7
                                          END IF
                                  CASE 17408 'F10
                                          QuitEntry% = TRUE
```

```
                                        Stops% = TRUE
                                CASE 97 TO 122
                                        AcceptChar% = TRUE
                                CASE 65 TO 90
                                        AcceptChar% = TRUE
                                CASE 48 TO 57
                                        AcceptChar% = TRUE
                                CASE 58 TO 59
                                        AcceptChar% = TRUE
                                CASE 34
                                        AcceptChar% = TRUE
                                CASE 39
                                        AcceptChar% = TRUE
                                CASE 44
                                        AcceptChar% = TRUE
                                CASE 46
                                        AcceptChar% = TRUE
                                CASE 32
                                        AcceptChar% = TRUE
                                CASE ELSE

'*****************************************
                                        '**
         **
                                        '**            Error Handling
Needed      **
                                        '**
         **

'*****************************************
                        END SELECT
                        IF col% = 9 AND i% = 20 THEN
                                Stops% = TRUE
                                ChangeCol% = TRUE
                        ELSEIF col% > 9 AND i% = 11 THEN
                                Stops% = TRUE
                                ChangeCol% = TRUE
                        END IF
                        IF AcceptChar% = TRUE THEN Stops% = TRUE
                LOOP UNTIL Stops% = TRUE
                Stops% = FALSE
                IF AcceptChar% = TRUE THEN
                        IF InsertOn% = TRUE THEN
                                IF i% > 1 AND i% < LEN(current$) + 1
THEN
                                        current$ = LEFT$(current$, i%
- 1) + Character$ + RIGHT$(current$, LEN(current$) - i% + 1)
                                ELSEIF i% = 1 AND i% < LEN(current$)
+ 1 THEN
                                        current$   =   Character$   +
RIGHT$(current$, LEN(current$) - i% + 1)
```

```
                        ELSEIF i% = 1 THEN
                                current$ = Character$
                        ELSE
                                current$ = LEFT$(current$, i%
- 1) + Character$
                        END IF
                ELSE
                        IF i% > 1 AND i% < LEN(current$) THEN
                                current$ = LEFT$(current$, i%
- 1) + Character$ + RIGHT$(current$, LEN(current$) - i%)
                        ELSEIF i% = 1 AND i% < LEN(current$)
THEN
                                current$   =   Character$   +
RIGHT$(current$, LEN(current$) - i%)
                        ELSEIF i% = 1 THEN
                                current$ = Character$
                        ELSE
                                current$ = LEFT$(current$, i%
- 1) + Character$
                        END IF
                END IF
                LOCATE row%, col% + 1, 1
                IF col% = 9 THEN
                        PRINT    current$;    SPACE$(19    -
LEN(current$))
                ELSE
                        PRINT    current$;    SPACE$(10    -
LEN(current$))
                END IF
                i% = i% + 1
                AcceptChar% = FALSE
        END IF
        IF FileLine% < 1 THEN FileLine% = 1
        IF FileLine% > 100 THEN FileLine% = 100
        SELECT CASE col%
                CASE 9
                        Star$(FileLine%) = current$
                CASE 32
                        RA$(FileLine%) = current$
                CASE 47
                        DEC$(FileLine%) = current$
                CASE 62
                        EPOCH$(FileLine%) = current$
                CASE ELSE

'*****************************************
                                '**
         **
                                '**          Error Handling
Needed       **
```

```
                                          '**
         **

'****************************************
        END SELECT
        '
        IF ChangeFileLine% <> FALSE THEN
                FileLine% = FileLine% + ChangeFileLine%
                IF FileLine% < 1 THEN FileLine% = 1
                IF FileLine% > 100 THEN FileLine% = 100
                ChangeFileLine% = FALSE
                SELECT CASE col%
                        CASE 9
                                current$ = Star$(FileLine%)
                        CASE 32
                                current$ = RA$(FileLine%)
                        CASE 47
                                current$ = DEC$(FileLine%)
                        CASE 62
                                current$ = EPOCH$(FileLine%)
                        CASE ELSE

'****************************************
                                          '**
         **
                                          '**          Error Handling
Needed      **
                                          '**
         **

'****************************************
                END SELECT
        END IF
        '
        IF HomeOrEnd% <> FALSE THEN
                IF HomeOrEnd% = 1 THEN
                        row% = FirstRow%
                ELSEIF HomeOrEnd% = 100 THEN
                        row% = LastRow%
                END IF
                FileLine% = HomeOrEnd%
                HomeOrEnd% = FALSE
                COLOR WHITE, BLUE, BLUE
                LOCATE LastRow%, 9, 0
                PRINT SPACE$(1)
                SELECT CASE col%
                        CASE 9
                                current$ = Star$(FileLine%)
                        CASE 32
                                current$ = RA$(FileLine%)
                        CASE 47
```

```
                                        current$ = DEC$(FileLine%)
                                CASE 62
                                        current$ = EPOCH$(FileLine%)
                                CASE ELSE

'*****************************************
                                        '**
         **
                                        '**             Error Handling
Needed      **
                                        '**
         **

'*****************************************
                        END SELECT
                END IF
                '
                IF ChangeCol% = TRUE THEN
                    SELECT CASE col%
                        CASE 9
                                col% = 32
                                current$ = RA$(FileLine%)
                        CASE 32
                                col% = 47
                                current$ = DEC$(FileLine%)
                        CASE 47
                                col% = 62
                                current$ = EPOCH$(FileLine%)
                        CASE 62
                                col% = 9
                                row% = row% + 1
                                FileLine% = FileLine% + 1
                                IF FileLine% < 1 THEN FileLine% = 1
                                IF FileLine% > 100 THEN FileLine% =
100
                                current$ = Star$(FileLine%)
                        CASE ELSE

'*****************************************
                                        '**
         **
                                        '**             Error Handling
Needed      **
                                        '**
         **

'*****************************************
                    END SELECT
                    i% = 1
                    ChangeCol% = FALSE
                END IF
```

```
 LOOP UNTIL QuitEntry% = TRUE
'
END SUB

'*************************************
'**     Name:        StarDisplay       **
'**     Type:        Subroutine        **
'**     Module:      MIRACTRL.BAS      **
'**     Language:    Quickbasic 4.50   **
'*************************************
'
'Creates the display screen graphics for star data entry
'
'Example of use:  StarDisplay
'Parameters:     (none)
'Variables:      i%      Looping index
'Module Level
'    DECLARATIONS:  DECLARE SUB StarDisplay
'
SUB StarDisplay
'
'Clear field
 COLOR WHITE, BLUE, BLUE
 FOR r% = 14 TO 21
        LOCATE r%, 6, 0
        PRINT SPACE$(68)
 NEXT r%
'
'Fill in star data entry fields
 COLOR WHITE, BLUE, BLUE
 LOCATE 14, 31, 0
 PRINT "Please enter stars"
 LOCATE 15, 33, 0
 PRINT "RA             DEC";
 LOCATE 16, 10, 0
 PRINT "Star Name                    hr:min:sec";
 LOCATE 16, 48, 0
 PRINT CHR$(248); ":':''";
 LOCATE 16, 63, 0
 PRINT "Epoch";
'
'Draw fields for star data input
 FOR i% = 1 TO 5
        COLOR WHITE, BLACK, BLUE
        LOCATE 16 + i%, 10, 0
        PRINT SPACE$(19)
        LOCATE 16 + i%, 33, 0
        PRINT SPACE$(10)
        LOCATE 16 + i%, 48, 0
        PRINT SPACE$(10)
        LOCATE 16 + i%, 63, 0
```

```
        PRINT SPACE$(10)
 NEXT i%
 LOCATE 23, 5, 0
 COLOR WHITE, BLUE, BLUE
 PRINT "F10 - SAVE and exit data entry"
'
END SUB

'****************************************
'**      Name:        TeleAlign        **
'**      Type:        Subroutine       **
'**      Module:      MIRACTRL.BAS     **
'**      Language:    Quickbasic 4.50  **
'****************************************
'
'Allows  user  input  of  the  telescope's  Hour  Angle  and
Declination to align
'the telescopein the local coordinate system.
'
'Example of use: TeleAlign GMST0hrSEC#, GHAaries#, TeleRA#,
TeleDEC#
SUB TeleAlign (GMST0hrSEC#, GHAaries#, TeleRA#, TeleDEC#)
'
'Initialize variables
 MIRAlatitude# = .63908139#        'radians, (36 deg 37 min
North)
 MIRAlongitude# = 2.12639281#     'radians, (121 deg 50 min
West)
 sec% = VAL(RIGHT$(TIME$, 2))
 min% = VAL(MID$(TIME$, 4, 2))
 hour% = VAL(LEFT$(TIME$, 2))
'
'Calculate Greenwhich Mean Sidereal Time
 'Determine the day fraction that has passed since midnight
 'in SECONDS
 sec& = sec%
 hours& = hour%
 mins& = min%
 dayFraction# = CDBL((hours& * 3600) + (mins& * 60) + sec&)
 '
 'Determine the Greenwhich Mean Sidereal Time at the current
 'time in SECONDS
 GMSTinSEC# = GMST0hrSEC# + dayFraction#
'
'Calculate Greenwhich Hour Angle of Aries
 GHAaries# = GMSTinSEC# * 2# * pi / 86164.09053099999#'Seconds
of solar time converted to radians
 DO WHILE GHAaries# > (2# * pi)
        GHAaries# = GHAaries# - (2# * pi)
 LOOP
 DO WHILE GHAaries# < 0#
```

```
        GHAaries# = GHAaries# + (2# * pi)
 LOOP
'
'Calculate the telescope's hour angle for this moment
 RAofMIRA# = GHAaries# - MIRAlongitude#
'
 '**********************************************
'Read telescope's Hour Angle from the encoders **
 'Store to variable HourAngle# in radians      **
 '**********************************************
 TeleRA# = RAofMIRA# - HourAngle#   'in radians
 DO WHILE TeleRA# > (2# * pi)
        TeleRA# = TeleRA# - (2# * pi)
 LOOP
 DO WHILE TeleRA# < 0#
        TeleRA# = TeleRA# + (2# * pi)
 LOOP
'
 '***********************************************
'Read telescope's Declination from the encoders **
 'Store to variable TeleDEC# in radians         **
 '***********************************************
'
END SUB

'**************************************
'**     Name:       TeleAngles       **
'**     Type:       Subroutine       **
'**     Module:     MIRACTRL.BAS     **
'**     Language:   Quickbasic 4.50  **
'**************************************
'
'Calculates the telescope's zenith and azimuth angles in the
'terrestrial coordinate frame.
'
'Example of use:  TeleAngles Zenith%, Azimuth%, GHAaries#,
RAofStar#, DECofStar#, CelCoord#(), DCM#(), TerraRec#()
'
SUB TeleAngles (Zenith%, Azimuth%, GHAaries#, RAofStar#,
DECofStar#, CelCoord#(), DCM#(), TerraRec#())
'
'Initialize variables
 MIRAlatitude# = .63908139#       'radians, (36 deg 37 min
North)
 MIRAlongitude# = 2.12639281#    'radians, (121 deg 50 min
West)
'
'Define a celestial coordinate vector in RECTANGULAR
coordinates
 CelCoord#(1) = COS(RAofStar#) * COS(DECofStar#)
 CelCoord#(2) = SIN(RAofStar#) * COS(DECofStar#)
```

```
 CelCoord#(3) = SIN(DECofStar#)
'
'Define the direction cosine matrix which relates celestial
and terrestrial coordinates
 DCM#(1, 1) = (-SIN(GHAaries#) * SIN(MIRAlongitude#) *
SIN(MIRAlatitude#) - COS(GHAaries#) * COS(MIRAlongitude#) *
SIN(MIRAlatitude#))
 DCM#(1, 2) = (-SIN(GHAaries#) * COS(MIRAlongitude#) *
SIN(MIRAlatitude#) + COS(GHAaries#) * SIN(MIRAlongitude#) *
SIN(MIRAlatitude#))
 DCM#(1, 3) = COS(MIRAlatitude#)
 DCM#(2, 1) = (-COS(GHAaries#) * SIN(MIRAlongitude#) +
SIN(GHAaries#) * COS(MIRAlongitude#))
 DCM#(2, 2) = (-SIN(GHAaries#) * SIN(MIRAlongitude#) -
COS(GHAaries#) * COS(MIRAlongitude#))
 DCM#(2, 3) = 0
 DCM#(3, 1) = (COS(GHAaries#) * COS(MIRAlongitude#) *
COS(MIRAlatitude#) + SIN(GHAaries#) * SIN(MIRAlongitude#) *
COS(MIRAlatitude#))
 DCM#(3, 2) = (SIN(GHAaries#) * COS(MIRAlongitude#) *
COS(MIRAlatitude#) - COS(GHAaries#) * SIN(MIRAlongitude#) *
COS(MIRAlatitude#))
 DCM#(3, 3) = SIN(MIRAlatitude#)
'
'Calculate the terrestrial rectangular coordinates
 TerraRec#(1) = DCM#(1, 1) * CelCoord#(1) + DCM#(1, 2) *
CelCoord#(2) + DCM#(1, 3) * CelCoord#(3)
 TerraRec#(2) = DCM#(2, 1) * CelCoord#(1) + DCM#(2, 2) *
CelCoord#(2) + DCM#(2, 3) * CelCoord#(3)
 TerraRec#(3) = DCM#(3, 1) * CelCoord#(1) + DCM#(3, 2) *
CelCoord#(2) + DCM#(3, 3) * CelCoord#(3)
'
'Calculate the azimuth and zenith angles
 IF TerraRec#(1) < 0# AND TerraRec#(2) < 0# THEN
       Azimuth# = ATN(TerraRec#(1) / TerraRec#(2)) + (pi /
2#)
 ELSEIF TerraRec#(1) >= 0# AND TerraRec#(2) < 0# THEN
       Azimuth# = ATN(-TerraRec#(2) / TerraRec#(1))
 ELSEIF TerraRec#(1) < 0# AND TerraRec#(2) >= 0# THEN
       Azimuth# = ATN(-TerraRec#(2) / TerraRec#(1)) + pi
 ELSE Azimuth# = ATN(TerraRec#(1) / TerraRec#(2)) + ((3# * pi)
/ 2#)
 END IF
 Azimuth% = INT(Azimuth# * 360# / (2# * pi))
 DO WHILE Azimuth% < 0
       Azimuth% = Azimuth% + 360
 LOOP
 DO WHILE Azimuth% > 360
       Azimuth% = Azimuth% - 360
 LOOP
```

```
 Elevation# = ATN(TerraRec#(3) / SQR(1 - TerraRec#(3) ^ 2#))
 Zenith% = 90 - INT(Elevation# * 360# / (2# * pi))
 '
END SUB

'****************************************
'**     Name:       TeleStatusDisplay **
'**     Type:       Subroutine        **
'**     Module:     MIRACTRL.BAS      **
'**     Language:   Quickbasic 4.50   **
'****************************************
'
'Creates the display screen for various elements of telescope
'status and updates them while the telescope is tracking or
slewing.
'
'Example of use:  TeleStatusDisplay ActiveStar$
'Parameters:    (none)
'Variables:      Star$()      A 5-element matrix of desired
stars for
'                            viewing
'                 RA$()        A 5-element matrix of right
ascensions
'                            for the stars listed in Star$()
'              DEC$()      A 5-element matrix of declinations
'                            for the stars listed in Star$()
'                 ActiveStar$  A string naming the star for
which a tracking
'                                solution is currently being
calculated
'                i%, r%      Looping indices
'Module Level
'          DECLARATIONS:     DECLARE  SUB  TeleStatusDisplay
(ActiveStar$)
'
SUB TeleStatusDisplay (ActiveStar$, Simulation%)
 CLS
'Color the background
 FOR i% = 1 TO 25
     COLOR BLUE, BLACK, BLUE
     PRINT STRING$(80, 177)
 NEXT i%
 LOCATE 24, 1, 0
 PRINT STRING$(80, 177);
 LOCATE 25, 1, 0
 PRINT STRING$(80, 177);
'
'Draw top edge of star name box
 COLOR WHITE, BLUE, BLUE
 LOCATE 10, 29, 0
 PRINT CHR$(201); STRING$(19, 205); CHR$(187);
```

```
COLOR WHITE, BLUE, BLUE
LOCATE 10, 38, 0
PRINT "Star"
'
'Draw the body of the star name box and place the name
COLOR WHITE, BLUE, BLUE
LOCATE 11, 29, 0
PRINT CHR$(186); STRING$(19, 0); CHR$(186);
LOCATE 11, INT((80 - LEN(ActiveStar$)) / 2), 0
PRINT ActiveStar$
'
'Draw bottom edge of the star name box
COLOR WHITE, BLUE, BLUE
LOCATE 12, 29, 0
PRINT CHR$(200); STRING$(19, 205); CHR$(188);
'
'Place the lettering
COLOR WHITE, RED, BLUE
LOCATE 2, 27, 0
PRINT " ESC key halts telescope "
COLOR WHITE, BLUE, BLUE
LOCATE 6, 28, 0
PRINT "Oliver Observing Station"
LOCATE 7, 32, 0
PRINT "Telescope Status"

'Draw top edge of status box
LOCATE 13, 5, 0
COLOR WHITE, BLUE, BLUE
PRINT CHR$(201); STRING$(68, 205); CHR$(187);
'
'Draw the body of the status box
FOR r% = 14 TO 21
   LOCATE r%, 5, 0
   PRINT CHR$(186);
   PRINT SPACE$(68);
   PRINT CHR$(186);
NEXT r%
'
'Draw bottom edge of the status box
LOCATE 22, 5, 0
PRINT CHR$(200); STRING$(68, 205); CHR$(188);
LOCATE 22, 23, 0
PRINT " ENTER to change telescope status "
'
'Place the help instructions
LOCATE 23, 31, 0
PRINT "F1 for help screen"
'Place the lettering
COLOR WHITE, BLUE, BLUE
LOCATE 14, 10, 0
```

```
 PRINT "RA"
 LOCATE 17, 10, 0
 PRINT "DEC"
 LOCATE 20, 10, 0
 PRINT "Hour Angle"
 LOCATE 14, 34, 0
 PRINT "Zenith Angle"
 LOCATE 17, 36, 0
 PRINT "Azimuth"
 LOCATE 14, 58, 0
 PRINT "PST"
 LOCATE 17, 58, 0
 PRINT "Sidereal Time"
 LOCATE 20, 58, 0
 PRINT "UTC"
'
'Show motor state boxes
 COLOR RED, BLUE, BLUE
 LOCATE 1, 35, 0
 PRINT "Simulation"
 COLOR WHITE, BLUE, BLUE
 LOCATE 5, 7, 0
 PRINT "RA Slew"
 LOCATE 5, 17, 0
 PRINT "RA Step"
 LOCATE 5, 57, 0
 PRINT "DEC Slew"
 LOCATE 5, 66, 0
 PRINT "DEC Step"
 FOR i% = 1 TO 2
        LOCATE 2, 10 * i% - 1, 0
        PRINT CHR$(201); CHR$(205); CHR$(187)
        LOCATE 3, 10 * i% - 1, 0
        PRINT CHR$(186); SPACE$(1); CHR$(186)
        LOCATE 4, 10 * i% - 1, 0
        PRINT CHR$(200); CHR$(205); CHR$(188)
 NEXT i%
 FOR i% = 1 TO 2
        LOCATE 2, (10 * i% - 1) + 50, 0
        PRINT CHR$(201); CHR$(205); CHR$(187)
        LOCATE 3, (10 * i% - 1) + 50, 0
        PRINT CHR$(186); SPACE$(1); CHR$(186)
        LOCATE 4, (10 * i% - 1) + 50, 0
        PRINT CHR$(200); CHR$(205); CHR$(188)
 NEXT i%
 '
END SUB

'****************************************
'**     Name:        TrackCommands      **
'**     Type:        Subroutine         **
```

```
'**     Module:     MIRACTRL.BAS       **
'**     Language:   Quickbasic 4.50    **
'****************************************
'
'Controls  the  right  ascension  stepping  motor  only  to
compensate
'for local coordinate drift due to the rotation of the Earth.
'
'Example of use:  TrackCommands
'
SUB TrackCommands
'
'Track as appropriate in Right Ascension only
'Update motor display box
 COLOR RED, BLUE, BLUE
 LOCATE 3, 20, 0
 PRINT CHR$(222)
'POKE (address of RA track controller), X
END SUB

'****************************************
'**     Name:       VideoState         **
'**     Type:       Subprogram         **
'**     Module:     BIOSCALL.BAS       **
'**     Language:   Quickbasic 4.50    **
'****************************************
'
'Determines the current video mode parameters.
'
'Example of use: VideoState mode%, columns%, page%
'Parameters:     mode%     Current video mode
'                columns%  Current number of text columns
'                page%     Current active display page
'Variables:      reg       Structure of type RegType
'MODULE LEVEL
'    DECLARATIONS:   TYPE RegType
'                        ax     AS INTEGER
'                        bx     AS INTEGER
'                        cx     AS INTEGER
'                        dx     AS INTEGER
'                        Bp     AS INTEGER
'                        si     AS INTEGER
'                        di     AS INTEGER
'                        flags AS INTEGER
'                    END TYPE
'
'   DECLARE SUB Interrupt (intnum%, inreg AS RegType, outreg
AS RegType)
'    DECLARE SUB VideoState (mode%, columns%, page%)
'
SUB VideoState (mode%, columns%, page%) STATIC
```

```
    DIM reg AS RegType
    reg.ax = &HF00
    Interrupt &H10, reg, reg
    mode% = reg.ax AND &HFF
    columns% = (CLNG(reg.ax) AND &HFF00) \ 256
    page% = (CLNG(reg.bx) AND &HFF00) \ 256
END SUB

'***************************************
'**     Name:          Windows          **
'**     Type:          Subprogram       **
'**     Module:        WINDOWS.BAS      **
'**     Language:      Quickbasic 4.50  **
'***************************************
'
'Displays a rectangular window for information display
'or menu selection.
'
'Example of use: Windows w1, wText$(), wTitle$, wPrompt$
'
'Parameters:       w1          Structure of type WindowsType
'                  wText$()    Array of strings to be displayed
'                  wTitle$     Title string
'                  wPrompt$    Prompt string
'
'Variables:        mode%          Current video mode
'                   columns%        Current number of character
columns
'                  page%          Current video page
'                  cursorRow%     Saved cursor row position
'                  cursorCol%     Saved cursor column position
'                  newpage%       Next video page
'                 lbText%        Lower boundary of array of text
lines
'                 ubText%        Upper boundary of array of text
lines
'                  i%             Looping index
'                   maxlen%         Length of longest string to
display
'                  length%        Length of each array string
'                   row2%            Row number at bottom right
corner of window
'                  col2%          Column number at bottom right
corner of
'                                 window
'                     ul%              Upper left corner border
character code
'                    ur%              Upper right corner border
character code
'                     ll%              Lower left corner border
character code
```

```
'                  lr%               Lower right corner border
character code
'                 vl%          Vertical border character code
'                  hl%            Horizontal border character
code
'                 r%            Index to each line of text
'                  choice$        Set of unique characters for
each menu line
'                 tmp$          Work string
'                kee%         Key code returned by InKeyCode%
function
'MODULE LEVEL
'   DECLARATIONS:  SUB Windows (w AS WindowsType, wText$(),
wTitle$,
'                              wPrompt$) STATIC
'
SUB windows (w AS WindowsType, wText$(), wTitle$, wPrompt$)
STATIC
' Key code numbers
  CONST DOWNARROW = 20480
  CONST ENTER = 13
  CONST ESCAPE = 27
  CONST UPARROW = 18432
'
' Determine current video page
  VideoState mode%, columns%, page%
'
' Record current cursor location
  cursorRow% = CSRLIN
  cursorCol% = POS(0)
'
' Window will be on the next page, if available
  newpage% = page% + 1
  IF newpage% > 7 THEN
     SCREEN , , 0, 0
     PRINT "Error: Windows - not enough video pages"
     SYSTEM
  END IF
'
' Copy current page to new page
  PCOPY page%, newpage%
'
' Show the current page while building window on new page
  SCREEN , , newpage%, page%
'
' Determine array bounds
  lbText% = LBOUND(wText$)
  ubText% = UBOUND(wText$)
'
' Check the text array bounds, lower always 1, upper > 0
  IF lbText% <> 1 OR ubText% < 1 THEN
```

```
      SCREEN , , 0, 0
      PRINT  "Error:  Windows  -  text  array  dimensioned
incorrectly"
      SYSTEM
   END IF
'
' Determine longest string in text array
   maxLen% = 0
   FOR i% = lbText% TO ubText%
      length% = LEN(wText$(i%))
      IF length% > maxLen% THEN
         maxLen% = length%
      END IF
   NEXT i%
'
' Determine the bottom right corner of window
   row2% = w.row + ubText% + 1
   col2% = w.col + maxLen% + 3
'
' Check that window fits on screen
   IF w.row < 1 OR w.col < 1 OR row2% > 25 OR col2% > columns%
THEN
      SCREEN , , 0, 0
      PRINT "Error: Windows - part of window is off screen"
      PRINT "columns% = "; columns%
      SYSTEM
   END IF
'
' Set the edge characters
   SELECT CASE w.edgeline
   CASE 0
      ul% = 32
      ur% = 32
      ll% = 32
      lr% = 32
      vl% = 32
      hl% = 32
   CASE 1
      ul% = 218
      ur% = 191
      ll% = 192
      lr% = 217
      vl% = 179
      hl% = 196
   CASE 2
      ul% = 201
      ur% = 187
      ll% = 200
      lr% = 188
      vl% = 186
      hl% = 205
```

```
  CASE ELSE
      SCREEN , , 0, 0
      PRINT "Error: Windows - Edge line types incorrect"
      SYSTEM
  END SELECT
'
'Draw top edge of box
 LOCATE w.row, w.col, 0
 COLOR w.fgdEdge, w.bgdEdge
 PRINT CHR$(ul%); STRING$(maxLen% + 2, hl%); CHR$(ur%);
'
'Draw the body of the window
 FOR r% = w.row + 1 TO row2% - 1
    LOCATE r%, w.col, 0
    COLOR w.fgdEdge, w.bgdEdge
    PRINT CHR$(vl%);
    COLOR w.fgdBody, w.bgdBody
    tmp$  =  LEFT$(wText$(r%  -  w.row)  +  SPACE$(maxLen%),
maxLen%)
    PRINT " "; tmp$; " ";
    COLOR w.fgdEdge, w.bgdEdge
    PRINT CHR$(vl%);
 NEXT r%
'
'Draw bottom edge of the box
 LOCATE row2%, w.col, 0
 COLOR w.fgdEdge, w.bgdEdge
 PRINT CHR$(ll%); STRING$(maxLen% + 2, hl%); CHR$(lr%);
'
'Center and print top title, if present
 IF wTitle$ <> "" THEN
    LOCATE w.row, (w.col + col2% - LEN(wTitle$) + 1) \ 2, 0
    COLOR w.fgdTitle, w.bgdTitle
    PRINT wTitle$;
 END IF
'
'Center and print prompt, if present
 IF wPrompt$ <> "" THEN
    LOCATE row2%, (w.col + col2% - LEN(wPrompt$) + 1) \ 2, 0
    COLOR w.fgdPrompt, w.bgdPrompt
    PRINT wPrompt$;
 END IF
'
'Now make the new page visible and active
 SCREEN , , newpage%, newpage%
'
'Take next action based on action code
 SELECT CASE w.action
 CASE 1
    'Get a key code number and return it
    DO
```

```
        w.returnCode = KeyCode%(Character$)
    LOOP UNTIL w.returnCode
 '
 CASE ELSE
    w.returnCode = 0
 END SELECT
'
'Reset the cursor position
 LOCATE cursorRow%, cursorCol%
'
END SUB

'****************************************
'**      Name:        WindowsPop         **
'**      Type:        Subprogram         **
'**      Module:      WINDOWS.BAS        **
'**      Language:    Quickbasic 4.50    **
'****************************************
'
'Removes last displayed window
'
'Example of use:WindowsPop
'Parameters:     (none)
'Variables:      mode%       Current video mode
'                columns%    Current number of display columns
'                page%       Current display page
'
'Module level
'   DECLARATIONS:  DECLARE SUB WindowsPop
'
SUB WindowsPop STATIC
    VidecState mode%, columns%, page%
    IF page% THEN
        SCREEN 0, , page% - 1, page% - 1
    END IF
END SUB
```

B. SAMPLE DATA FILE "STARFILE.DAT"

```
"Sirius","23:01:40","00:00:00","2000"
"Betelguese","03:05:00","00:00:00","2000"
"Pollux","12:34:21","23:45:11","2000"
"Dubhe","250:34:59","58:34:59","2000"
"Castor","14:11:00","55:32:10","2000"
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
"","","",""
```

LIST OF REFERENCES

1. *The Astronomical Almanac*, Nautical Almanac Office, United States Naval Observatory, Washington, D.C., and Her Majesty's Nautical Almanac Office, Royal Greenwich Observatory, London, England, 1990 and 1992.

2. Bowditch, Nathaniel, *American Practical Navigator*, Vols I and II, Defense Mapping Agency Hydrographic/Topographic Office Publication No. 9, 1984.

3. Hergert, Douglas, *Microsoft QuickBASIC Programmer's Reference*, Howard W. Sams and Company, 1989.

4. Craig, John C., *Microsoft QuickBASIC Programmer's Toolbox*, Microsoft Press, 1988.

5. Green, Robin M., *Spherical Astronomy*, Cambridge University Press, 1985.

6. Skilling, Hugh H., *Fundamentals of Electric Waves*, Robert E. Krieger Publishing Company, 1942.

7. Bollinger, John G. and Duffie, Neil A., *Computer Control of Machines and Processes*, Addison Wesley, 1989.

8. Ratcliffe, J.A., *An introduction to the ionosphere and magnetosphere*, Cambridge University Press, 1972.

9. Danby, J.M.A., *Fundamentals of Celestial Mechanics*, 2nd Ed., Willmann-Bell, 1988.

10. Halliday, David and Resnick, Robert, *Fundamentals of Physics*, 3rd Ed., Wiley, 1988.

11. Lang, K.R., *Astrophysical Formulae*, 2nd Ed., Springer-Verlag, Berlin, 1980.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library Code 0052 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code AA Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5000	1
4.	Monterey Institute for Research in Astronomy ATTN: Bruce Weaver, Ph.D. Director of Research 900 Major Sherman Lane Monterey, California 93940-5000	2
5.	Department of Aeronautics and Astronautics ATTN: Professor I.M. Ross, Code AA/Ro Naval Postgraduate School Monterey, California 93943-5000	1
6.	Lieutenant David P. Wood 270-58-4588 Student, Naval Dive and Salvage Training Center Panama City, Florida 32407-5002	2